Integrated Berth Allocation and Quay Crane Assignment Problem: Set partitioning models and computational results

Iris, Cagatay; Pacino, Dario; Røpke, Stefan; Larsen, Allan

Published in:
Transportation Research. Part E: Logistics and Transportation Review

Link to article, DOI:
10.1016/j.tre.2015.06.008

Publication date:
2015

Document Version
Peer reviewed version

Integrated Berth Allocation and Quay Crane Assignment Problem: Set Partitioning Models and Computational Results

Çağatay Iris*, Dario Pacino*, Stefan Ropke**,Allan Larsen*

*Technical University of Denmark, Department of Transport**

*Technical University of Denmark, Department of Management Engineering***

Abstract

Most of the operational problems in container terminals are strongly interconnected. In this paper, we study the integrated berth allocation and quay crane assignment problem in seaport container terminals. We will extend the current state-of-the-art by proposing novel set partitioning models. To improve the performance of the set partitioning formulations, a number of variable reduction techniques are proposed. Furthermore, we analyze the effects of different discretization schemes and the impact of using a time-variant/invariant quay crane allocation policy. Computational experiments show that the proposed models significantly improve the benchmark solutions of the current state-of-art optimal approaches.

Index terms— Container terminal operations; Berth allocation problem; Quay crane assignment; Mixed integer programming; Set partitioning; Column reduction

1. Introduction

Fierce global competition in production and trade has forced all entities in supply chains to optimize their logistics operations. The never ending quest for shorter lead times and reduced cost for logistics cost requires extremely efficient logistics systems. In 2013, the worldwide container trade accounts for about 22% of the 6.7 billion tons of dry-cargo trade, and all loads are being transported by vessels via container terminals (UNCTAD [27]). Recent statistics show that total container trade volumes reached 160 million Twenty-foot Equivalent Units (TEUs) in 2013 with a growth of 4.6% (UNCTAD [27]). These statistics suggest that logistics efficiency heavily relies on effective container terminal operations. Due to the increasing importance of container terminals and high complexity of their operations, the need for optimization has become evident in recent years. This can also be perceived by the increase of scientific literature where operations research techniques are applied to container terminals (see Stahlbock and Voß [24] for a survey).

Recent advances in the modeling of container terminal problems have pushed the focus towards integration issues. Conventional hierarchical optimization of sequential operations is known to have possible disadvantages, that may result in infeasible, suboptimal or poor solutions. This is because decisions made in earlier steps were made without considering the resultant knock-on effects in the following stages. This paper focuses on two important problems on the quayside of terminal operations: the Berth Allocation Problem (BAP) and the Quay Crane Assignment Problem (QCAP). The first problem allocates berthing positions and times for vessels, while the second determines the number of quay cranes (QCs) to be assigned for the load and discharge operations. These two problems are mutually dependent. The number of available QCs depends on where and when the vessel is berthed. Concurrently, the berthing time depends on the processing time of the vessel, which in turn depends on the number of cranes assigned. An integration of those two problems was first introduced by Park and Kim [20].

The goal of our work is to solve the integrated Berth Allocation and quay Crane Assignment Problem (BACAP), where a berthing time and position for each vessel is assigned during a given planning horizon. A solution to the problem also includes the assignment of QCs, by factoring in marginal productivity losses due to crane interference, and processing times depending on the berthing position of the vessel. An objective is to propose a method that solves instances to optimality. When instances cannot be solved to optimality, tight upper and lower bounds on the objective should be generated. Such bounds can be used to evaluate the performance of future and past heuristics.

An important factor in berth and QC management is the use of policies for the assignment of QCs to vessels. Hence why this paper analyses two main policies: time-invariant and time-variant QC assignment. The first policy decides how many QCs to assign to a given vessel, and this number cannot change throughout the stay at berth. The second relaxes this assumption and allows the number of assigned QCs to vary during the ship's stay at port. In both cases the number of QCs assigned lies within a given interval specified by the contract between the terminal and the shipping companies. Both variants of the BACAP are modeled in this paper using a Generalized Set Partitioning (GSPP) formulation. Furthermore, a set of column reduction techniques are presented which help to limit the number of feasible columns generated and to provide better bounds.

The literature on the BAP distinguishes between discrete and continuous versions of the problem with respect to the berth partitioning. In the former version, vessels can only berth at predefined sections of the quay, while this restriction does not apply to the latter version. The work proposed in this paper addresses optimal approaches for the continuous case; Where the berth space is discretized in the same manner as Meisel and Bierwirth [18], Meisel and Bierwirth [19] and Turkogullari et al. [25], with berthing at integer points (e.g. every 10 meters). Three discretization techniques are tested in this paper however only one of the them guarantees optimal solutions to the original problem.

In order to validate and evaluate our models, we present a comparison with the BACAP state-of-the-art results by Meisel and Bierwirth [18]. In Meisel and Bierwirth [18] a compact mathematical model, which can optimally solve some instances of up to 20 vessels, is presented. For larger instances (30, 40 vessels), the model cannot generate integer upper bounds. In the same paper, these upper bounds are generated using different heuristic approaches.

This paper presents three major contributions: 1) Novel generalized set partitioning formulations that can solve more instances than previous models. 2) Improved upper and lower bounds to almost all instances, bounds that will be of use when testing new algorithms for the problem. 3) Techniques for reducing the number of variables in the model, techniques that can be useful for problems that use a similar modeling approach.

The paper is organised as follows: First, a literature review is presented in Section 2. In Section 3, the problem definition and mathematical models proposed by Meisel and Bierwirth [18] are given. The proposed GSPP models and column reduction techniques are presented in Section 4. Extensive computational results are presented and discussed in Section 5. The paper is concluded by Section 6.

2. Literature Review

The importance of container terminal problems has been revealed by many academic studies where authors illustrate recent trends and point out gaps in the literature (see Stahlbock and Voß [24] for a general review). As regards the integrated quayside problems, recent surveys of Bierwirth and Meisel [1], Bierwirth and Meisel [2] focus on berth allocation and QC planning problems (assignment and scheduling) in container terminals. Authors classify berth allocation problems according to spatial, temporal, processing time, and performance indicator attributes. Integration of berth allocation and QC assignment is classified as deep, hierarchical or through a feedback loop. Most papers present compact formulations with deep integration. The BAP remains the main problem and the additional problem is either the assignment or scheduling of QCs. Recently, Meisel and Bierwirth [19] integrated three of the main seaside terminal planning problems, i.e. BAP, QCAP (in numbers and specific QC assignment) and the quay crane scheduling problem (QCSP).

The BAP is classified as static or dynamic with respect to whether the arrival time of the vessels imposes a bound on the berth start time. One of the first models for dynamic BAP was presented by Imai et al. [11] and was successively improved by Imai et al. [12] for the continuous berth allocation case. The latter presents a two-stage heuristic approach which uses discrete berthing solutions and reallocates them in a continuous manner. Cordeau et al. [8] proposed a Tabu Search (TS) for the dynamic discrete BAP and a continuous variant. A well performing simulated annealing approach is proposed by Kim and Moon [14] for the continuous BAP.

2.1. BACAP literature - BAP, QCAP properties

A list of relevant literature for the BACAP is summarized in Table 1, in which information about the problem structure, objective function and solution approaches are presented. The studies are listed in chronological order of publication year. In the pioneering paper for the BACAP, Park and Kim [20] presented a model for the problem. The model supports time-variant QC assignments and is solved by using lagrangian relaxation-based heuristics. Afterwards a dynamic programming method assigns the specific QCs to vessels. With respect to spatial attributes, some papers focus on discrete berth allocation in the BACAP (Imai et al. [13], Giallombardo et al. [9], Vergados et al. [29]). However, continuous berth layout in the BACAP has also attracted many researchers (see Table 1). Different extensions appear in the literature surrounding the berth allocation properties of BACAP. Meisel and Bierwirth [18] considered the marginal productivity losses due to the QC interference. Experiments with different levels of congestion show the strong impact of the QC-interference on the cost function. The handling time which depends on the berthing position is modeled by Meisel [17]. Another extension is the modeling of operational constraints of QCs. Giallombardo et al. [9] proposed a QC profile scheme in which the authors include the effects of shifts, the interference of QCs, the priority of vessels and various real-life constraints. They proposed a two-stage heuristic. In the first stage, QC profiles are assigned to each vessel. In the second stage, authors solve the remaining BAP via a TS heuristic. The BACAP is also studied by Blazewicz et al. [3]. They considered the problem as a parallel machine scheduling problem and seek to minimize the makespan.

Problem variations can also be found with respect to the QC assignment. The two main policies are the time-variant and time-invariant QC assignment. In the time-invariant version, authors mostly solve the QC assignment problem first and then solve the BAP (Liang et al. [15], Chen et al. [6], etc.). Another modeling aspect is whether individual QCs are assigned or the number of QCs to serve each vessel is determined. Imai et al. [13] considered the assignment of specific QCs through detailed QC movement constraints. This ensures the assignment of specific QCs, however, the relationship between the number QCs deployed and the processing time could be improved. In another example, Chen et al. [6] made

specific QC-to-vessel assignment and this facilitates the calculation of QC requirements. They proposed valid inequalities that link the berth scheduling and QC assignment better and some valid inequalities are in the form of non-crossing constraints.

2.2. BACAP literature - Objective function properties

In terms of cost function, we see variations in the modeling of the BACAP. The most popular objective (see Table 1) is the composition of berthing costs (QC costs) and time-dependent penalty costs (Chang et al. [5], Raa et al. [21], Meisel and Bierwirth [18], etc.). Total weighted service time is another popular objective of the formulations (Liang et al. [15], Yang et al. [30], etc.). As mentioned in Section 2.1, the deviation from expected berthing position may be embedded in the objective with a cost (Chang et al. [5], Raa et al. [21], Turkogullari et al. [25]). Instead of being in the objective, the deviation from expected berthing position might be modeled to affect the processing time (as in Meisel and Bierwirth [18]). Then the model becomes harder to solve, because the processing time of a vessel would not only depend only on the load of vessel which is mostly a parameter, it would also depend on a decision variable which is the berthing position.

2.3. BACAP literature - Solution techniques

The solution approaches are clustered in novel mathematical models, exact methods and heuristic/analytic methods in Table 1. Most of the papers propose novel mathematical models for the variants of BACAP. Raa et al. [21] enrich current models by taking vessel priorities, preferred berthing positions and QC-assignment-dependent handling times into account. The proposed BACAP model is solved using a rolling horizon approach.

2.3.1. Exact Methods

Several authors use set partitioning formulations to solve different quayside planning problems. The first use of GSPP aimed at solving the BAP (Christensen and Holst [7]), where the authors proposed a branch-and-price algorithm. The approach can be used to solve both discrete and continuous BAP. For instances of up to 35 vessels, optimal solutions can be found for the discrete BAP, while a gap of 8.3% is obtained for the continuous version. Buhrkal et al. [4] generated columns a priori and solved the same GSPP model for the discrete case with an IP solver. The approach clearly improved the state-of-the-art and solved the BAP up to 60 vessels to optimality. A recent study by Saadaoui et al. [23] also focuses on the discrete BAP in which 10 berths are considered. They solve a linear programming (LP) relaxation of GSPP model using column generation. When the column generation terminates, they impose the integrality constraints again and resolve the GSPP with the active pool of columns. The framework solves instances of 120 vessels with an average optimality gap of 0.20%. Umang et al. [26] proposed a GSPP model to solve a more complicated variant of BAP with hybrid berth layout in bulk ports. The model, with a priori generated columns, can solve instances up to 40 vessels to optimality. Robenek et al. [22] formulated a GSPP model for the integrated berth allocation and yard assignment problem in bulk ports. The problem considers the cargo types on the vessel which affect the storage location in the yard and consequently the berth allocation. They solve the problem with a branch-and-price algorithm. The instances include 10 cargo locations in the yard and 10 berths are available with different equipment. The authors solve instances with 10, 25, and 40 vessels with average optimality gaps of 0.37%, 4.11% and 3.76%, respectively. The branch-and-price is only run for instances of 10 vessels. Due to the time complexity, instances with 25 and 40 vessels are solved with the column generation and the integrality constraints are imposed in the last stage of column generation to obtain an upper bound.

Vacca et al. [28] establish the first decomposition method for BACAP and it is based on the model by Giallombardo et al. [9]. The authors suggest a QC profile which holds productivity losses due to QC interferences, vessel priorities and QC assignment for each shift. The authors have implemented a branch-and-price scheme and several accelerating techniques. The approach obtains the optimal results for 10 and 15 vessels and an average gap of 2.95% is obtained for 20 vessels and 5 berths in three hours of time limit.

An exact method to solve the BACAP with continuous (but discretized for each 50 meter) berth layout is presented by Turkogullari et al. [25]. The authors solely consider a time-invariant QC assignment policy. They first formulate a mathematical model to solve BACAP. The model generates optimum solutions up to 60 vessels where there are 24 berthing sections. In addition to that, the authors propose a cutting plane algorithm to solve the BACAP with specific QC-to-vessel assignment by using the optimum solutions of original BACAP model. It is noted that cutting plane algorithm can convert each optimum BACAP solution to the optimum solution of BACAP with specific QC-to-vessel assignment for the instances which are tested.

Chen et al. [6] have proposed a Benders decomposition method over the berth-level model proposed by Liu et al. [16]. The authors model a reduced version of the BACAP, where the berthing position of each vessel is given, thus leaving the berthing start/end times, specific QC-to-vessel assignment and the positions of QCs as the only decisions to make. The model is decomposed into a master problem and a sub-model, and the results show that the decomposition technique is faster than the original formulation.

In this study, we present exact methods to solve variants of the BACAP (presented in Meisel [17] and Meisel and Bierwirth [18]). Let us now clarify the differences between the BACAP definition used in this paper and that of Vacca et al. [28] and Turkogullari et al. [25]. The problems considered in Christensen and Holst [7], Buhrkal et al. [4] and

Saadaoui et al. [23] do not take QC assignment into account, while Chen et al. [6] assume a partial berth assignment is given. In Vacca et al. [28], the authors formulate the BACAP with discrete berth allocation, QC moves from one vessel to another are only allowed at the end of the working shifts (restricted time-variant QC assignment). The authors do not consider berthing position dependent processing times, and there are also some differences due to the QC profile definition and the objective function formulation. In Turkogullari et al. [25], the problem is solely considered for time-invariant QC assignment case of the BACAP. The marginal productivity losses due to the QC interference are not taken into account. The same is true for the berthing deviation dependent processing times and speeding up option. In their paper, the authors also focus on specific QC-to-vessel assignment problem.

2.3.2. Heuristic Algorithms

There are also heuristic/analytic approaches which try to solve BACAP. The most popular metaheuristic used to solve BACAP is genetic algorithms (GAs). Imai et al. [13], Liang et al. [15], Yang et al. [30] etc. test various GA configurations which are specific for the defined problem. In the paper by Liang et al. [15], three different chromosome structures are used to prioritize vessels, to allocate berth, and to assign QC numbers. Chang et al. [5] formulate the chromosome as a composition of four-dimensional indice pertaining to the arrival sequence, berthing position, berthing time and number of QCs for vessels. Meisel and Bierwirth [18] propose three heuristics to solve the problem. They conclude that squeaky wheel optimization (SWO) along with local refinements does slightly better than TS. They also show that SWO and TS are better than the First Come First Served-based heuristic. Vergados et al. [29] propose a Constraint Programming (CP) model with a tailored branching heuristic within a Large Neighborhood Search framework. The model includes QC-to-vessel assignment considering gang (a team of operators that work on QCs) allocations.

The work presented in this paper builds on the model presented by Meisel and Bierwirth [18]. The literature survey and Table 1, show that the model incorporates many relevant constraints and includes several aspects of the problem in its objective function. We therefore believe that the model is a good starting point for the studies carried out in our paper.

3. Modeling the BACAP: Meisel and Bierwirth (2009) Model

Before going into the details of the solution approach, let us introduce the BACAP. We do so by presenting the formulation proposed by Meisel and Bierwirth [18] and its time-invariant QC assignment version. We propose an extension to this model that allows modeling problems where the number of QCs assigned to a vessel cannot change during the vessel's stay at the berth.

The objective of the BACAP is to find the best berthing position and time for upcoming vessels by fulfilling the QC requirement of each vessel. A solution for each vessel determines the berthing position, the berthing start, end times and the number of QCs that are operating on the vessel at any given time. The time horizon is discretized. Any discretization can be used but it is useful to think of a discretization into whole hours. The berthing position is determined by a continuous variable, but the data used with the model ensures that ships are berthed at integer positions.

An example of a BACAP plan can be seen in Figure 1 that shows the berthing plan in a time/space diagram. In this example, three ships are berthed in the harbor. Each vessel is represented by a rectangle that shows the time and space occupied by the vessel. The smaller rectangles indicate the assignment of QCs to vessels, each small rectangle represents one QC. Each ship has an upper and lower limit on the number of cranes that can be assigned to it. These bounds are determined by contracts between the vessel owner and the port and by the size of the ship. A limited number of QCs are available in the harbor and this determines the maximum number of cranes that we can assign in any time slot. The symbols used on the Figure 1 will be explained in the following section.

Table 1: Integrated BACAP literature abstract

		Problem Structure												Objective Function								Solution Approach													
		A			B		C	D	E	F		G										X	Y			Z									
Year	Authors	1	2	3	1	2	Y	Y	Y	1	2	1	2	Type	T_w	E	T	L	Δb	C_{qc}	C_k	1	1	2	3	1	2	3	4	5	6	7			
2003	Park and Kim [20]	*				*				*		*	*	Min		*		*	*	*		*	*												
2008	Imai et al. [13]		*		*				*	*		*	*	Min	*							*				*									
2009	Liang et al. [15]		*		*						*		*	Min	*											*									
2009	Meisel and Bierwirth [18]		*			*	*	*		*			*	Min		*	*	*		*		*					*	*	*						
2010	Giallombardo et al. [9]		*		*		*		*	*			*	Max						-*	*	*						*							
2010	Chang et al. [5]		*			*	*		*	*		*	*	Min		*	*		*	*						*				*					
2011	Raa et al. [21]			*		*	*			*			*	Min				*	*	*		*													
2011	Blazewicz et al. [3]	*				*			*	*	*	*	*	Min	*																	*			
2012	Yang et al. [30]		*			*					*	*	*	Min	*					*		*				*									
2012	Chen et al. [6]		*			*					*	*	*	Min				*							*										
2013	Vacca et al. [28]		*		*		*		*	*			*	Max						-*	*			*											
2013	Vergados et al. [29]		*		*				*	*		*	*	Min				*	*	*		*								*					*
2014	Turkogullari et al. [25]		*			*			*		*	*	*	Min			*		*			*			*										

Problem Structure = {A: Temporal attribute: (1: Static, 2: Dynamic, 3: Rolling horizon), B: Spatial attribute: (1: Discrete, 2: Continuous), C: Interference, D: Deviation depending handling times, E: Operational constraints: Columns with "Y" heading means "Yes, the attribute is taken into account".
F: QC assignment: (1: Time-variant, 2: Time-invariant), G: QC policy: (1: QC-to-vessel, 2: The number of QCs to assign)}
Objective Function = {T_w: Total weighted service time (handling, waiting, etc.), E: Earliness, T: Tardiness, L: Lateness, Δb: Deviation from expected berthing position, C_{qc}: Cost of QC assignment or changing QC-plan, -* represents gains earned through QC plans, C_k: Cost of housekeeping or other operations}
Solution Approaches = {X:1: Novel mathematical models, Y: Exact Methods: (1: Lagrange relaxation, 2: Branch-and-Price, 3: Other exact methods), Z: Heuristic, Simulation, AI approaches: (1: Genetic algorithms, 2: Squeaky wheel optimization, 3: Tabu search, 4: Greedy/LNS heuristics, 5: Simulation, 6: Approximation heuristics, 7: Constraint programming}

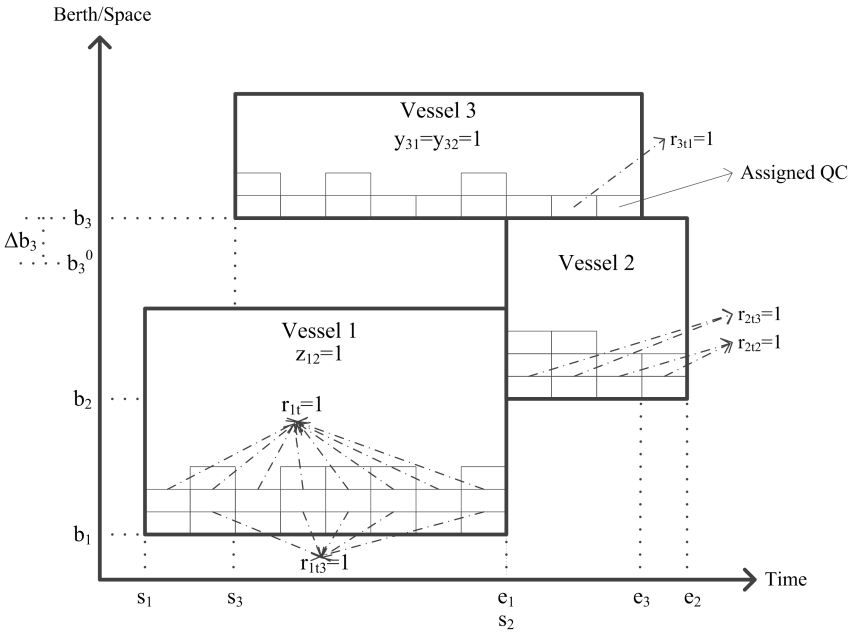

Figure 1: Berth-time diagram of BACAP.

The objective function is a combination of time-dependent costs and QC assignment costs. The time dependent costs can be attributed to the berthing start and end times while the QC dependent costs are a function of how many QCs are assigned to each vessel. If a vessel is berthed before its Expected Time of Arrival (ETA), a speed-up cost occurs for each rushed time unit. Delay costs depend on how many time units have passed from the Expected Finishing Time (EFT). An ulterior penalty cost incurs if the berthing end time is beyond the Latest Finishing Time (LFT). An example of the cost structure of vessel 1 can be seen in Figure 2. The figure shows that during the vessel's stay, between its start (s_1) and end time (e_1), QC operations costs are distributed along periods depending on the number of QCs operating. Since the vessel start time is four periods earlier than its ETA (ETA_1), speed-up costs occur. In the same manner, we must pay delay costs due to the vessel's end time being later than the EFT (EFT_1). Finally, since operations go beyond the LFT (LFT_1), a one-time penalty is added to the cost.

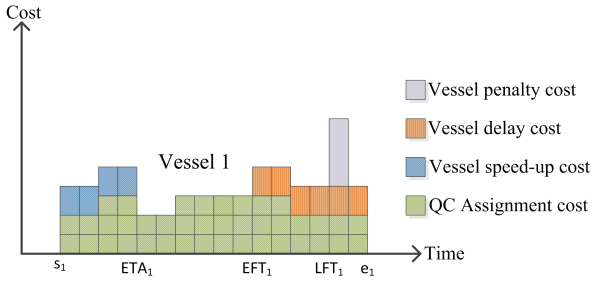

Figure 2: Cost structure of BACAP for Vessel 1 in Figure 1.

3.1. Time-variant model

Let us now introduce the model proposed by Meisel and Bierwirth [18]. Table 2 presents the mathematical notation.

Table 2: BACAP mathematical notation

	Parameters and sets:
V	Set of all vessels to be served, $V \in \{1, 2, .., N\}$, where N is the number of vessels to be planned
L	Length of the quay
T	Set of 1 hour periods, $T \in \{0, 1, .., H-1\}$, where H is the end of planning horizon
l_i	Length of vessel $i \in V$
b_i^0	Desired berthing position of vessel $i \in V$
m_i	Quay crane capacity demand of $i \in V$ given as QC-hours
r_i^{min}	Minimum number of QCs agreed to serve vessel $i \in V$ simultaneously
r_i^{max}	Maximum number of QCs allowed to serve vessel $i \in V$ simultaneously
R_i	Feasible range of QCs assignable to vessel $i \in V$, where $R_i = [r_i^{min}, r_i^{max}]$
ETA_i	Expected arrival time of vessel $i \in V$
EST_i	Earliest time of arrival of vessel $i \in V$ when it is speed up
EFT_i	Expected finishing time of vessel $i \in V$
LFT_i	Latest finishing time of vessel $i \in V$ without any penalty cost
c_i^1	Speed up cost of vessel $i \in V$ on its journey to catch a berthing time earlier than ETA_i
c_i^2	Cost of exceeding the expected finishing time EFT_i for vessel $i \in V$
c_i^3	Penalty cost by exceeding LFT_i for vessel $i \in V$
c^4	Cost rate per QC-hour of operations
α	Interference exponent for the QCs. Only q^α effective QC hours are obtained when assigning q QCs to a ship for one hour
β	Berth deviation factor. A ship i placed at position b_i needs $(1 + \lvert b_i^0 - b_i \rvert \beta) m_i$ effective QC hours. $\lvert b_i^0 - b_i \rvert$ is the deviation from desired berthing position
M	A large positive number
Q	Available number of QCs
	Decision variables:
$b_i \in \mathbb{Z}^+$	Berthing position of vessel $i \in V$
$s_i \in \mathbb{Z}^+$	Time of starting the handling (berthing start time) of vessel $i \in V$
$e_i \in \mathbb{Z}^+$	Time of ending the handling (berthing end time) of vessel $i \in V$
$r_{it} \in \mathbb{B}$	1; if there is any QC assignment to vessel i in period t, 0 otherwise
$r_{itq} \in \mathbb{B}$	1; if there is exactly q QC assigned to vessel i in period t, 0 otherwise
$\Delta b_i \in \mathbb{Z}^+$	Deviation from desired berth if vessel i is in position b, $\Delta b_i = \lvert b_i^0 - b_i \rvert$
$\Delta ETA_i \in \mathbb{Z}^+$	Required speedup to reach start-time s_i by vessel i, where $\Delta ETA_i = \lvert ETA_i - s_i \rvert$
$\Delta EFT_i \in \mathbb{Z}^+$	Tardiness of vessel $i \in V$ when operations are finished later than expected finishing time, $\Delta EFT_i = \lvert e_i - EFT_i \rvert$
$u_i \in \mathbb{B}$	1; if finishing time of vessel $i \in V$ exceed latest finishing time, 0 otherwise
$y_{ij} \in \mathbb{B}$	1; if vessel i is berthed below vessel j in berth area, i.e. $b_i + l_i \leq b_j$, 0 otherwise
$z_{ij} \in \mathbb{B}$	1; if handling of vessel i ends no later than handling of vessel j starts in berth area, 0 otherwise
	Time invariant decision variables:
$p_{iq} \in \mathbb{B}$	1; if q QCs are assigned to vessel i, 0 otherwise

$$\min \sum_{i \in V} (c_i^1 \Delta ETA_i + c_i^2 \Delta EFT_i + c_i^3 u_i + c^4 \sum_{t \in T} \sum_{q \in R_i} r_{itq} q) \tag{1}$$

subject to

$$\sum_{t\in T}\sum_{q\in R_i} q^{\alpha} r_{itq} \geq (1+\Delta b_i \beta) m_i \quad \forall i \in V \tag{2}$$

$$\sum_{i\in V}\sum_{q\in R_i} q r_{itq} \leq Q \quad \forall t \in T \tag{3}$$

$$\sum_{q\in R_i} r_{itq} = r_{it} \quad \forall i \in V, \forall t \in T \tag{4}$$

$$\sum_{t\in T} r_{it} = e_i - s_i \quad \forall i \in V \tag{5}$$

$$(t+1) r_{it} \leq e_i \quad \forall i \in V, \forall t \in T \tag{6}$$

$$r_{it} t + H(1-r_{it}) \geq s_i \quad \forall i \in V, \forall t \in T \tag{7}$$

$$\Delta b_i \geq b_i - b_i^0 \quad \forall i \in V \tag{8}$$

$$\Delta b_i \geq b_i^0 - b_i \quad \forall i \in V \tag{9}$$

$$\Delta ETA_i \geq ETA_i - s_i \quad \forall i \in V \tag{10}$$

$$\Delta EFT_i \geq e_i - EFT_i \quad \forall i \in V \tag{11}$$

$$Mu_i \geq e_i - LFT_i \quad \forall i \in V \tag{12}$$

$$b_j + M(1-y_{ij}) \geq b_i + l_i \quad \forall i,j \in V, \quad i \neq j \tag{13}$$

$$s_j + M(1-z_{ij}) \geq e_i \quad \forall i,j \in V, \quad i \neq j \tag{14}$$

$$y_{ij} + y_{ji} + z_{ij} + z_{ji} \geq 1 \quad \forall i,j \in V, \quad i \neq j \tag{15}$$

$$s_i, e_i \in \{EST_i, \dots H\} \quad \forall i \in V \tag{16}$$

$$b_i \in \{0, 1, \dots L - l_i\} \quad \forall i \in V \tag{17}$$

$$\Delta ETA_i, \Delta EFT_i \geq 0 \quad \forall i \in V \tag{18}$$

$$r_{itq}, r_{it}, u_i, y_{ij}, z_{ij} \in \{0,1\} \quad \forall i,j \in V, \forall t \in T, \forall q \in R_i \quad i \neq j \tag{19}$$

The objective function (1) is a minimization of the overall cost which has two major components. The first, is based on the vessels' time at port (speed-up cost, delay cost, and penalty cost). The second, is linked to the QC assignments in which the number of QCs used is multiplied by the cost rate per QC-hour. Constraint (2) ensures that every vessel receives the required QC capacity, taking into account productivity losses by QC interference, and increased QC demand due to deviation from the expected berthing position. Constraint (3) enforce restrictions so that the assigned QC number cannot exceed the available number of QCs in the terminal. Constraint (4) links the r_{itq} and r_{it} variables: if any q QCs are assigned to vessel i in period t, then operations are ongoing on the vessel which should therefore stay at berth. Constraints (5), (6) and (7) link the r_{it} variables with the arrival and departure variables. The constraints guarantee that the r_{it} variables are only set to one when $t \in [s_i, e_i]$ and that operations are not preemptive. Constraint (8)-(12) determine the deviation from the expected berthing place, the required speed-up for vessel to reach s_i, the tardiness of the operations, and sets u_i to one if the ship departs after LFT_i. Constraint (13) and (14) are used to set the variables y_{ij} and z_{ij}. These variables are used in constraint (15) to avoid that ships overlap in time or space. Definition of domains (16) and (17) ensure that start-time and end-time of operations are between the ETA_i and the end of the planning horizon. The berthing position of vessel i is restricted by the berth and the vessel length. Constraints (18) and (19) define the domains of the remaining variables.

3.2. Time-invariant model

In the model presented in Section 3.1 the number of QCs assigned to a vessel can change over time. However, some terminals may opt not to change the number of QCs throughout the vessel's stay at port, in order not to create additional congestion of QC rescheduling, and therefore we propose a variant of the model where the number of QCs assigned to a vessel is fixed throughout the vessel's stay. Mind that the number of cranes to assign is still a decision variable. This problem has been studied in academic literature and is named time-invariant BACAP (Turkogullari et al. [25], Yang et al. [30], Meisel [17], etc.). To model the time-invariant QC assignment we add the binary decision variable p_{iq} which is one iff q QCs are assigned to vessel i. The time-invariant QC assignment is then enforced by the following constraints:

$$\sum_{q\in R_i} p_{iq} = 1 \quad \forall i \in V \tag{20}$$

$$r_{itq} \leq p_{iq} \quad \forall i \in V, \forall t \in T, \forall q \in R_i \tag{21}$$

$$p_{iq} \in \{0,1\} \quad \forall i \in V, \forall q \in R_i \tag{22}$$

Constraint (20) ensures that exactly one QC number is chosen for each vessel i. Constraint (21) links the r_{itq} and p_{iq} variables. If q QCs are assigned to vessel i ($p_{iq} = 1$), r_{itq} is either one or zero. We have to allow $r_{itq} = 0$ since there are some periods t where the vessel is not at berth. If p_{iq} equals zero the corresponding r_{itq} are forced to zero through the entire planning horizon. Constraint (5) guarantees avoiding preemption by preventing any zero values for r_{itq} within the berthing interval. Constraint (20), along with Constraints (5-7), ensure that only a fixed number of QCs is used without preemption in operations.

4. Generalized Set Partitioning Formulations

In this section, we present GSPP reformulations for the time-variant and time-invariant BACAP. These models are based on models for the berth allocation problem presented by Christensen and Holst [7] (see also Buhrkal et al. [4]). The addition of QC decisions is, to the best of our knowledge, novel. The proposed models contain a large number of variables and it is tempting to use column generation to solve them. However, in this paper we use the simpler approach of generating all variables a priori (as it also was successfully done in Buhrkal et al. [4]).

4.1. Time-Invariant GSPP Model

In the time-invariant GSPP model, a column represents a feasible assignment of a single vessel to a position in time and space (recall Figure 1), as well as an assignment of QCs for the berthing period. In addition to the already introduced notation, we introduce some additional notation. The set of columns (assignments) is denoted by Ω. We define three matrices (a_{ij}),(b_{pj}) and (q_{tj}), all containing $|\Omega|$ columns. Matrix (a_{ij}) contains a row for each vessel. Each element a_{ij} is binary and it is 1 iff column j represents an assignment of vessel $i \in V$. Each element of (a_{ij}) contains exactly one non-zero element. Binary matrix (b_{pj}) contains a row per (berth, time) position. The entry b_{pj} is one iff position $p \in P$ is occupied in the assignment that variable y_j represents. The matrix (q_{tj}) contains a row per time unit. An element q_{tj} indicates the number of QCs that are assigned to vessel j in time period t. Since we are modeling the time-invariant version of the problem, each column contains zeroes and one or more copies of a number $\tilde{q}$ which indicate the number of QCs used in the assignment that the variable represent. Each column j has a cost c_j. This cost is easily calculated from the vessel index, the (berth, time) position and the QC allocation. In the GSPP model, the berth dimension is discretized into S berth cells. Each vessel can occupy multiple cells when the discretization is fine enough. We let P be the set of (berth, time) positions that a ship can occupy. This set contains $H \cdot S$ elements. The decision variables of the models are denoted $y_j, j \in \Omega$, they are binary and indicate whether column (assignment) j should be used in the solution. The model is:

$$\min \sum_{j \in \Omega} c_j y_j \tag{23}$$

subject to

$$\sum_{j \in \Omega} a_{ij} y_j = 1 \qquad \forall i \in V \tag{24}$$

$$\sum_{j \in \Omega} b_{pj} y_j \leq 1 \qquad \forall p \in P \tag{25}$$

$$\sum_{j \in \Omega} q_{tj} y_j \leq Q \qquad \forall t \in T \tag{26}$$

$$y_j \in \{0,1\} \quad \forall j \in \Omega \tag{27}$$

The objective function (23) minimizes the sum of the costs for the selected variables. Constraint (24) guarantees that all vessels are served. Constraint (25) restricts each berth/time position to be used at most once. Constraint (26) ensures that we do not use more QCs than are available at the container terminal.

We illustrate the model with a small example containing two vessels, the first with a length of one and the second with a length of two berth units. Vessels 1 and 2 have a requirement of 2 and 4 QC hours, respectively. In this example we disregard that interference and a bad positioning can increase QC capacity demand. Furthermore, Vessel 1 has $\{r_i^{min}, r_i^{max}\} = \{1, 2\}$, and vessel 2 has $\{r_i^{min}, r_i^{max}\} = \{3, 5\}$. The earliest berthing start times (EST_i) for the two vessels are 1 and 2. Additionally, we assume that there are two berthing spaces, three planning periods, and six QCs available to serve the vessels. For the first vessel, all feasible solutions are presented in Table 3, while for the second, only a small portion is illustrated. The first two rows indicate which vessel the column is representing. The next six rows represent the 6 available time/space positions and indicate which position each assignment occupies. The last three rows indicate how many QCs are used in each time period by the assignment. The 15 columns with heading y_j indicate 15 possible assignments for vessel 1 and 2 while the column RHS gives the right hand side of each constraint. The last column simply

Table 3: Structure of assignment matrix for given example

$j=$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	RHS	
Vessel 1	1	1	1	1	1	1	1	1	1	1						$=1$	a_{ij}
Vessel 2											1	1	1	1	1	$=1$	
Space1/Time1	1				1											≤ 1	b_{pj}
Space1/Time2	1	1				1					1	1		1		≤ 1	
Space1/Time3		1					1				1		1		1	≤ 1	
Space2/Time1			1					1								≤ 1	
Space2/Time2			1	1					1		1	1		1		≤ 1	
Space2/Time3				1						1	1		1		1	≤ 1	
Time 1	1		1		2			2								≤ 6	q_{tj}
Time 2	1	1	1	1		2			2		3	4		5		≤ 6	
Time 3		1		1			2			2	3		4		5	≤ 6	

indicates the mathematical representation (symbol) of the columns in the model. Note that some of the columns presented in Table 3 might be removed by the column reduction techniques which will be discussed later.

The simple structure of the model is convenient, but its drawback is that the model can contain many variables (dependent on choice of planning horizon and discretization of the berth space). This model also handles the case where the number of QCs assigned to a vessel vary from time-period to time-period. This variant can be represented by allowing the entries in each column of the (q_{tj}) matrix to take more than two values.

Modeling the time-variant QC assignment this way, however, will increase the number of variables dramatically, therefore we have not pursued this direction. Instead, a different modeling approach for the time-variant number of QCs is presented in Section 4.2.

4.2. Time-variant GSPP Model

As for the time-invariant model, a column for the time-variant GSPP formulation represents a feasible assignment of a single vessel to a berth with its expected processing time. The difference, compared to the model from Section 4.1, is on how the processing times and QC assignment are handled. The exact number of QCs to serve the vessel in each period is not embedded in the column representation. Alternatively, since we know the minimum and maximum number of QCs that can serve a vessel in parallel (r_i^{min}, r_i^{max}), we can calculate the minimum and maximum processing time for a given ship at a given position (recall that position impacts processing time though the β parameter). Then, we proceed to generate an assignment for each possible processing time. The set of columns is again denoted by Ω. We define two matrices (a_{ij}), (b_{pj}) which contain $|\Omega|$ columns. a_{ij} and b_{pj} are interpreted in the same way as in Section 4.1. $\Omega^B(b,i)$ is the set of columns that places the start of ship i in berth b (so a column will only occur in one of the sets $\Omega^B(b,i)$ even if it takes up several berths). $\Omega^T(t,i)$ is the set of columns (assignments) that represent a placement of ship i that occupies time period t. Each column j has a cost value c_j. This cost includes the cost components which are related to the timing of the vessel (too early/too late), but leaves out the component related to the number of QCs used (see (1)), since this information cannot be deduced from the information in the column. There are two sets of decision variables: y_j determines if the column j is used or not, while r_{itq} is a binary variable that is 1 if q cranes are assigned to vessel i at time t. The additional parameters and notations that are not listed in Section 3 and 4.1 are as follows:

Additional set notations for GSPP model:
P: Set of positions: a position is a pair (berth, time slot), $P \in \{0, 1, .., H \cdot S\}$
B: Set of berthing spaces, $B \in \{1, 2..., S\}$
$\Omega^B(b,i)$: The set of columns representing a placement of ship i in berth b.
$\Omega^T(t,i)$: The set of columns representing a placement of ship i that occupies time period t.
Additional parameters:
$\underline{t}_{i,b}$: Minimum time (number of time periods needed to serve ship i in berth b): $\underline{t}_{i,b} = \left\lceil \frac{(1+\beta\Delta b_{ib})m_i}{(r_i^{max})^\alpha} \right\rceil$
$\bar{t}_{i,b}$: Maximum time (number of time periods needed to serve ship i in berth b): $\bar{t}_{i,b} = \left\lceil \frac{(1+\beta\Delta b_{ib})m_i}{(r_i^{min})^\alpha} \right\rceil$
Δ_{bi}: The absolute distance of berthing place b from desired position of vessel i: $\Delta_{bi} = \lvert b_i^0 - b \rvert$
Decision variables:
$y_j \in \{0,1\}$: 1 if column j (a certain ship/position/duration combination) is used, 0 otherwise
$r_{itq} \in \{0,1\}$: 1 if q cranes are assigned to ship i at time t, 0 otherwise

Hence, the mathematical model can be formulated as:

$$\min \sum_{j\in\Omega} c_j y_j + c_4\Big(\sum_{\forall i\in V}\sum_{\forall t\in T}\sum_{q\in R_i} qr_{itq}\Big) \tag{28}$$

subject to

$$\sum_{j\in\Omega} a_{ij} y_j = 1 \qquad \forall i \in V \tag{29}$$

$$\sum_{j\in\Omega} b_{pj} y_j \leq 1 \qquad \forall p \in P \tag{30}$$

$$\sum_{i\in V}\sum_{q\in R_i} qr_{itq} \leq Q \qquad \forall t \in T \tag{31}$$

$$\sum_{t\in T}\sum_{q\in R_i} q^{\alpha} r_{itq} \geq \sum_{b\in B}\left((1+\Delta_{bi}\beta)\sum_{j\in\Omega^B(b,i)} y_j\right) \qquad \forall i \in V \tag{32}$$

$$\sum_{q\in R_i} r_{itq} = \sum_{j\in\Omega^T(t,i)} y_j \qquad \forall i \in V, t \in T \tag{33}$$

$$y_j \in \{0,1\} \qquad \forall j \in \Omega \tag{34}$$

$$r_{itq} \in \{0,1\} \qquad \forall i \in V, t \in T, q \in R_i \tag{35}$$

The objective (28) is formulated as a sum of column costs (which include speedup, lateness and penalty costs) and QC assignments costs. Constraint (29) and (30) are the same as (23)-(27). Constraint (31) guarantees that at most Q QCs are used in each time period. The next two constraints (32) and (33) link the columns used and the QC assignment variables in terms of berthing time and position. Constraint (32) ensures that enough QC capacity is assigned to each vessel. The left hand side of constraint (32) measures the number of effective QC hours assigned to the vessel and takes the interference factor into account. The right hand side calculates how many effective QC hours are necessary and takes the berthing position into account. Constraint (33) imposes that QCs are only assigned to a vessel when it is at port. This constraint also imposes that at most one QC assignment policy can be applied ($\sum_{q\in\bar{Q}} r_{itq} \leq 1$ where $\bar{Q} \in \{1, 2..., Q\}$) for each vessel in each period. What is more, constraint (33) guarantees that the QC assignment is non-preemptive in the periods between the column start and end interval.

In the time-variant model, not only the number of constraints, but also the number of columns is higher compared to the time-invariant model. In the time-invariant GSPP, we typically generate $r_i^{max} - r_i^{min}$ columns for each ship and position (berth/time) combination. For the time-variant case it is $\bar{t}_{i,b} - \underline{t}_{i,b}$. Since the QC requirements (m_i) are rather high in the instances, we typically have $\bar{t}_{i,b} - \underline{t}_{i,b} > r_i^{max} - r_i^{min}$ and the time-variant GSPP thus needs a larger number of columns.

The GSPP models offer some modeling advantages compared to compact models like the one presented in Section 3.1. GSPP models allow the handling of many types of constraints implicitly while generating the feasible columns. Also various objective functions can easily be handled as long as they can be decomposed into a cost per column.

4.3. Discretization policies

The complexity of GSPP and the number of columns vary by using different discretization policies of the continuous berth space. In the original data set proposed by Meisel and Bierwirth [18], the length of the berth is 100 units (1000m: 100x10m segments). In this paper, three approaches are proposed to test the performance of the formulations:

- Berth Length of 1: In this representation, we have 100 berthing spaces (S) where each of them has a 1 unit of length (l_s : 10m in real-life). This formulation directly corresponds to the version studied by Meisel and Bierwirth [18].
- Berth Length of 2: In this version, we have 50 berthing spaces (S) where each of them has 2 units of length (l_s : 20m in real-life). This representation results in a smaller model but the solution quality is decreased since we cannot use the quay space as efficiently as in the BL=1 approach. This discretization is inspired by the distance between bollards at the port.
- Dynamic (Hybrid) Discretizing: In this approach, we let the discretization length be dependent on the specific vessel. The policy is derived from the observation that the berthing of the optimal solution for vessel i usually lies around its desired berthing position (b_i^0). Hence, we do a finer discretization for 5 berthing spaces around the desired berthing position, for the rest of the berth length, a discretization of 2 is used. This policy usually results in around 55 berthing spaces depending on whether b_i^0 is close to the start or the end of the berth.

The different discretization policies are tested in Section 5.3 for both the time-variant and time-invariant case.

4.4. Column Reduction Strategies and Valid Inequalities for set partitioning models

The number of necessary columns may be very large when dealing with a high number of vessels and a fine discretization. Hence, in this section we propose some rules for eliminating columns that cannot be part of the optimal solution. This decreases memory consumption and makes the model easier to solve.

In each column reduction technique, an upper bound (UB) $\bar{z}$ is required for the value of the objective function (1). By having this bound, we can decide whether to keep a column or simply remove it. The upper bound can be obtained using a heuristic for the BACAP, for now it is simply assumed that an upper bound is known.

4.4.1. Preprocessing-1: Simple redundancy

Given the upper bound $\bar{z}$ on the objective value, a simple but nevertheless useful preprocessing rule is to remove columns with cost $c_j > \bar{z}$. This applies to both GSPP models. But for the time-variant version, a better bound can be obtained since c_j does not include the QC component. To do so, we calculate a lower bound (LB) on the costs of the QC assignments. First, the minimum number of crane hours needed (θ) is calculated by (36).

$$\theta = \sum_{i \in V} r_i^{min} \left\lceil \frac{m_i}{\left(r_i^{min}\right)^{\alpha}} \right\rceil \tag{36}$$

Given a vessel, the shortest possible processing time (when r_i^{min} vessels are assigned) can be calculated with $\left\lceil \frac{m_i}{\left(r_i^{min}\right)^{\alpha}} \right\rceil$, we then multiply this with the ship's minimum number of required QCs. This is obviously a lower bound on the number of QC hours needed for the ship and it is easy to calculate. Using θ, we calculate a lower bound on the QC component of the objective using $\underline{z} = c^4\theta$ and all columns with $c_j + \underline{z} > \bar{z}$ can be removed.

4.4.2. Preprocessing-2: Contribution regarding Lower Bound (LB)

The second preprocessing procedure is based on calculating a lower bound on the total objective by selecting the "best" column for each ship. For each ship we calculate the increased lower bound caused by selecting a column j instead of the vessel's best column. If that lower bound is greater than the upper bound, column j can be discarded. In the following, the idea is explained in more detail. We first describe the procedure for the time-invariant case, since this is the simplest.

Let $\Omega(i)$ be the columns corresponding to vessel i, then we can calculate σ_i, the lowest column cost for columns representing ship i by:

$$\sigma_i = \min_{j \in \Omega(i)} \{c_j\} \tag{37}$$

A lower bound on the overall objective is then:

$$\underline{z}^2 = \sum_{i \in V} \sigma_i \tag{38}$$

Let $\tau(j)$ be the vessel associated with column j, then any column j for which $\underline{z}^2 + c_j - \sigma_{\tau(j)} > \bar{z}$ can be removed. The left hand side (LHS) computes the lower bound on the objective if column j is used instead of the best column for ship $\tau(j)$.

The preprocessing rule also works for the time-variant GSPP, but in this case it can be improved since c_j does not contain the QC component. Let $\epsilon(i, d, \Delta b)$ be a lower bound on the number of QC hours needed to serve ship i when berthed Δb units away from the desired position and having a stay of d time units at port. With this we can calculate an improved lower bound $\phi(j)$ for column j's contribution to the objective function:

$$\phi(j) = c_j + c_4\epsilon(\tau(j), d(j), \Delta b(j)) \tag{39}$$

where $d(j)$ and $\Delta b(j)$ are the duration of the port stay and the deviation from best berth position for column j, respectively. We now use $\phi(j)$ to define the lowest contribution σ_i for each ship i:

$$\sigma_i = \min_{j \in \Omega(i)} \{\phi(j)\} \tag{40}$$

and we compute the lower bound on the total objective as before: $\underline{z}^2 = \sum_{i \in V} \sigma_i$. A column can now be eliminated if $\underline{z}^2 + \phi(j) - \sigma_{\tau(j)} > \bar{z}$.

What remains, is to describe how we calculate the lower bound $\epsilon(i, d, \Delta b)$. When a vessel is placed Δb positions away from the desired position, we have to put in $(1 + \Delta b\beta)m_i$ raw crane hours to serve the vessel. To minimize the number of QC hours needed, we have to spread the work evenly during vessel's stay interval to avoid high interference factors. We would have to work for d^1 hours with x QCs and for d^2 hours with $x + 1$ QCs. A method to calculate the ϵ function is presented in Algorithm 1.

The idea behind the procedure is identifying available capacity gaps in given processing times. By knowing the processing time of a given vessel i, we can calculate how many periods corresponds to which number of QCs in a solution.

Algorithm 1 Approximation of ϵ

Require: $i, r_i^{min}, r_i^{max}, d(j), (1+\beta\Delta b(j))m_i$
  if $(d(j)(r_i^{min})^\alpha \geq (1+\beta\Delta b(j))m_i)$
    return $d(j)r_i^{min}$;
  else
    Find $q \in \{r_i^{min}, ..., r_i^{max}\}$ such that $d(j)q^\alpha \leq (1+\beta\Delta b(j))m_i \leq d(j)(q+1)^\alpha$
    $p = d(j)$;
    while $(p \geq 0)$ **do**
      $\delta = p(q+1)^\alpha + (d(j)-p)(q)^\alpha$;
      if $(\delta \geq (1+\beta\Delta b(j))m_i)$
        $result = p(q+1) + (d(j)-p)q$;
      end if
      $p = p-1$;
    end while
    return *result*;

Since it is a lower bound calculation procedure, the result shows the least amount of QC hours required in the given circumstances. We can now calculate q. If q was allowed to be fractional we would need to solve it by (41).

$$d(j)\hat{q}^\alpha = (1+\beta\Delta b(j))m_i \quad \Rightarrow \quad \hat{q}^\alpha = \frac{(1+\beta\Delta b(j))m_i}{d(j)} \quad \Rightarrow \quad (\hat{q}^\alpha)^{1/\alpha} = \left(\frac{(1+\beta\Delta b(j))m_i}{d(j)}\right)^{1/\alpha}$$

$$\hat{q} = \left\lfloor \left(\frac{(1+\beta\Delta b(j))m_i}{d(j)}\right)^{1/\alpha} \right\rfloor \tag{41}$$

Theorem: The number of quay crane hours calculated using the Algorithm (1) is a lower bound on the number of QC hours needed to serve vessel i when berthed Δb units away from the desired position and having a stay of d time units at the port.

Proof:

See Appendix.A for proof □

Corollary: There is always a QC assignment plan which only includes $\hat{q}$ or $\hat{q}+1$ number of QCs for each period (in which vessel i is at port), and this plan satisfies total QC requirement of vessel i (i.e. $(1+\Delta b\beta)m_i$) and minimizes the total number of QC hours needed to serve vessel i when berthed Δb units away from the desired position.

Proof:

Proven Theorem guarantees Corollary, see Appendix.A for proof of Theorem □

4.4.3. Probing-1: Feasible assignment set fixing

We classify the next two methods as *probing methods* since they fix the value of one variable to one and analyze the immediate consequences. If as a consequence the lower bound rises above the upper bound then the corresponding variable can be eliminated. These methods are explained using the notation for the time-variant GSPP, but they work just as well for the time-invariant version.

The procedure goes through all variables y_j and iteratively fixes them to one. Fixing a variable to one usually implies that many other variables (columns) are becoming infeasible due to the overlap in berth/time space. Let $I(j)$ be the set of infeasible columns when column j is used. A lower bound for the total cost when having selected j is:

$$\underline{z}^3(j) = \phi(j) + \sum_{i \in V \setminus \tau(j)} \left(\min_{j' \in \Omega(i) \setminus I(j)} \{\phi(j')\} \right) \tag{42}$$

the formula uses the cost lower bound of column j and adds the best cost of the remaining ships' columns. Taking into account that columns infeasible with the selection of column j are not included in the calculation, we have that if $\underline{z}^3(j)$ turns out to be greater than $\bar{z}$ then column j can be removed.

4.4.4. Probing-2: Vessel pairs fixing

The second probing method extends the previous method by considering pairs of vessels when computing lower bounds. The method starts by pairing vessels. First the variable that assigns ship j with lowest $\phi(j)$ is found. Among these N assignments the ones that overlap the most in time/berth space are selected to form the first pair. These assignments are removed from the set of available assignments and another pair is formed by selecting the ones with most overlap among the remaining assignments. This continues until all vessels are paired up (or one vessel remains). Now that vessels have

been paired up, we can compute a lower bound on the contribution of each vessel pair. For a vessel pair $\{i_1, i_2\}$ this is done by (43).

$$\underline{z}(i_1, i_2) = \min_{j_1 \in \Omega(i_1), j_2 \in \Omega(i_2) \setminus I(j_1)} \{\phi(j_1) + \phi(j_2)\} \tag{43}$$

i.e. by selecting an assignment j_1 for vessel i_1 and an assignment j_2 for vessel i_2 that minimizes $\phi(j_1) + \phi(j_2)$ and do not overlap. Let $\mathcal{P}$ be the set of pairs and assume N is even. A lower bound for the total objective is

$$\underline{z}^4 = \sum_{\{i_1, i_2\} \in \mathcal{P}} \underline{z}(i_1, i_2) \tag{44}$$

we again go through all $j \in \Omega$ and fix y_j to one, iteratively. Fixing y_j to one has several effects. The ship $\tau(j)$ corresponding to column j is part of exactly one pair from $\mathcal{P}$. Since j is fixed we may have to redo our choice for that pair. This amounts to finding the best assignment j' for the other ship in the pair while ensuring that assignments j and j' do not overlap. For the other pairs we check if the best assignment for that pair overlaps with j. If not, we go on and use the best assignment, if there is an overlap we search for the best assignment pair that does not overlap with j.

Let i be the ship that was paired up with $\tau(j)$ then we can write the lower bound obtained by fixing $y_j = 1$ formally as:

$$\underline{z}^4(j) = \phi(j) + \min_{j' \in \Omega(i) \setminus I(j)} \{\phi(j')\} + \sum_{\{i_1, i_2\} \in \mathcal{P} \setminus \{i, \tau(j)\}} \left(\min_{j_1 \in \Omega(i_1) \setminus I(j), j_2 \in \Omega(i_2) \setminus (I(j) \cup I(j_1))} \{\phi(j_1) + \phi(j_2)\} \right) \tag{45}$$

We can eliminate y_j whenever $\underline{z}^4(j) > \bar{z}$. The two probing algorithms are rather time consuming, but the running time can be kept at a reasonable level by careful implementation. The two simpler preprocessing routines are also executed before running the probing methods in order to reduce the set of available columns.

4.4.5. Valid inequality based on $\phi(j)$

The computed $\phi(j)$ bounds give rise to a simple inequality that eliminates some non-optimal solutions from the solution space:

$$\sum_{j \in \Omega} \phi(j) y_j \leq \bar{z} \tag{46}$$

Constraint (46) ensures that the sum of all lower bounds of columns cannot be larger than the upper bound. The inequality can cut away feasible integer solutions, but only those that have an objective greater than the upper bound. It is likely that a black-box solver will be able to generate cover inequalities from the inequality since it is a knapsack constraint.

4.4.6. Reduction of r_{itq} variables

Finally, the time-variant version of GSPP may be improved with respect to variables r_{itq}. There can be no QC assignment before the EST of vessels and assignment of QCs have to be within the given interval $R_i = \left[r_i^{min}; r_i^{max}\right]$. Constraints (47) and (48) eliminate QC assignments that do satisfy these requirements.

$$r_{itq} = 0 \quad \forall i \in V,\, q \notin R_i, t \in T \tag{47}$$

and

$$r_{itq} = 0 \quad \forall i \in V,\, q \in R_i, t \in T :\, t < EST_i \tag{48}$$

The preprocessing and probing described above remove some y_j variables. This can force some r_{itq} to zero. We let it be up to the preprocessing routines of the black-box IP solver to eliminate such r_{itq} variables.

4.5. Discussion of solution methods

In this paper we generate the complete models (23-27 for the time-invariant case and 28-35 for the time-variant case) using the columns that are left after the column reduction techniques presented in Section 4.4. These models are then solved by CPLEX.

Since the models contain a large number of columns, an alternative solution approach would be to solve the LP relaxation of the models using a column generation algorithm and obtain integer solutions using a branch-and-price algorithm. Such an approach has been used for related problems by, for example, Vacca et al. [28] and Robenek et al. [22]. A simpler alternative to branch-and-price is to solve the integer model in the last iteration using the columns generated while solving the LP relaxation of the complete model using a column generation algorithm. Such an approach is used by Saadaoui et al. [23], but it is not guaranteed to obtain an optimal solution when the problem is solved in this way.

It is not clear if a branch-and-price approach would be advantageous for the size of BACAP instances currently used in the literature (see e.g. Meisel and Bierwirth [18]) since CPLEX in general is very good at solving GSPP as long as the

model fits into memory. However for larger instances branch-and-price algorithms will be competitive considering that at some point the number of generated columns for the complete models simply becomes too large to fit in the memory (see Saadaoui et al. [23], for evidence of this for the BAP). The most interesting research direction related to branch-and-price algorithms is perhaps to use model 23-27 to solve the time-variant case (see comments at the end of Section 4.1) since we expect that the LP relaxation of the time-variant version of model 23-27 would be tighter than that of 28-35. Generating all columns for this model variant (23-27) is out of the question for all but the smallest instances, but its LP relaxation could be solved using column generation and a branch-and-price algorithm would therefore be feasible.

5. Computational Results

We compare our results to those that can be obtained by the model presented by Meisel and Bierwirth [18]. All models are solved by using the CPLEX 12.6 solver. The column generator and reduction techniques are implemented in C++. In order to have a fair comparison, the model presented by Meisel and Bierwirth [18] is run for 10 hours using our computer and CPLEX version. The best result from the original and our re-implementation are reported. All tests are run on a 32 core AMD Opteron at 2.8Ghz and 132Gb of RAM. All running times are measured in seconds. The running times are reported for both the column generation and solver times. Due to memory restrictions only 5 threads are active per experiment.

5.1. Benchmark instances and running conditions

The benchmark is provided by Meisel and Bierwirth [18]. The data set includes three main vessel types (namely Feeders, Medium, and Jumbo vessels). Furthermore each vessel type differs in technical specifications and cost values. The generation of these instances is based on empirical data. The benchmark consists of 30 instances, and contains ten instances of 20, 30, and 40 vessels, respectively. We consider a container terminal with a quay of length L=1000 meters with 10 QCs available. The planning horizon is one week (168 hours), and planning operations are based on working hours. It should be noted that the planning horizon is set as a hard constraint by the benchmarks.

The vessels' specifications and parameters regarding the arrival and finishing time and the cost values can be obtained from Meisel and Bierwirth [18]. The interference coefficient (α) is set to 0.9, and the increase in the QC-hours needed due to berthing deviation is set to 0.01 (Meisel and Bierwirth [18]).

The complete column generation procedure works as follows: First, all feasible columns are generated, and after that the two preprocessing techniques are applied. The probing methods described in Section 4.4.4 are run last, since they are the most time consuming and therefore it is beneficial to reduce the set of columns as much as possible before running them.

The models based on Meisel and Bierwirth [18] (time-variant and time-invariant versions) are rerun with the same conditions reported in their paper. CPLEX 12.6 is run with a time limit of 36000 seconds using the options: *emphasize optimality* and *aggressive cut generation*. It is observed that the compact models require the *aggressive cut generation* option since the computation for the root node relaxation takes only little time, and most of the time is spent on branching for an integer solution.

For the set partitioning formulations, we observed that the best results were obtained by setting the MIP emphasis parameter to *discover hidden feasible solutions* and by turning on *local branching heuristic*. Another strategy that is applied to overcome the problem of finding an integer initial solution is to warm-start the models. Such solutions are also necessary for providing an upper bound for the column reduction strategies. Section 5.2 explains how the warm start solutions are found.

5.2. Upper bound and warm start strategies

Warm starts (and consequently the upper bounds) are obtained by solving a simpler version of the GSPP model. For the time-invariant GSPP, modeling each berth with a length of 4 units provides a warm start for all versions of time-invariant models (berth length of 1 (BL=1), berth length of 2 (BL=2), and dynamic discretization). This simpler model is solved with a time limit of 15 minutes and, in most cases, the model can be solved to optimality.

For the time-variant version one can use the solution from the time-invariant model as long as the discretization policy is kept the same, e.g. the solution to the time-invariant GSPP (BL=1) is not an upper bound for the dynamic-discretized time-variant GSPP. The time-invariant version of the model, which will generate a warm start, is run with a time limit of 20 minutes (not including the time to obtain BL=4 results). The additional runtime depends on the computational time generating the upper bound of the time-invariant case. In small and medium scale instances, the upper bounds obtained from the time-invariant models perform quite well. However, for large scale instances, BL=1 time-invariant models do not perform well. Hence, for the BL=1 versions the upper bound is selected among the time-invariant version with a berth length of 2 or dynamic discretization.

5.3. Computational Results

We present results for both the time-invariant and time-variant versions of the GSPP models and compare the results with those provided by the model of Meisel and Bierwirth [18]. Moreover we analyze the impact of the three berth discretizations and the column reduction techniques.

The performance of GSPP models are presented in Tables 4-to-7. In each table, the first column, "#", indicates instance ID. The columns denoted "Z" show the best upper bounds obtained, while "LB" reports the best lower bounds found. The gap (G) is calculated between upper and lower bounds. In Tables 4 and 7, the "T_C" and "T_{OPT}" are the time spent (in seconds) generating columns and the time spent solving the mathematical model, respectively. The column "R_+^-" illustrates whether the optimal solution is found in the root node relaxation. If yes, there is a "+", otherwise a "-". The column R_{LB} reports the lower bounds obtained in the root node. It should be noted that if the instance is solved to optimality in the root node, there is no root node LB. Additionally, the number of nodes in the branch and bound (B&B) tree is presented in "#nodes B&B" columns. The following four columns show the effect of the column reduction techniques. The column under column reduction section named $\mid \Omega \mid$ shows the number of columns generated. After that, the number of columns left is reported in each cell. Column $\mid \Omega_1 \mid$ shows the number of columns left after the the two simple preprocessing steps, while $\mid \Omega_2 \mid$and $\mid \Omega_3 \mid$, shows the number of columns left after the probing methods 1 and 2, respectively. The two columns ("UB T_{OPT}" and "Z_{UB}") report the upper bound used as a warm start solution and the length of time we spent computing this upper bound. Different from Table 7, Table 4 (for the time-invariant case) contains results that show the performance without using the preprocessing steps in the last four columns.

5.3.1. Time-Invariant GSPP results

The computational results for the time-invariant version of the GSPP are shown in Table 4 and 6. There is a clear tradeoff between the number of columns and solution quality.

Table 4 shows the results for a berth length of 1 unit (l_s : 10m in real-life, BL=1) which is the original problem studied in Meisel [17]. In this case, the GSPP formulation produces optimal results for all small and medium scale instances ($N = 20$, 30 vessels). For large scale instances ($N = 40$ vessels), only four instances cannot be solved to optimality within the 10 hour time limit. For small and medium scale instances, the runtime (T_{OPT}) is always less than 13 minutes, and the optimal solution is often found in the root node. This is clearly not the case for the instances with large scale instances.

Note that the column generation time (T_C) is small for any type of instance. In most small and medium scale instances, it is less than 10 seconds. The number of generated columns can be reduced significantly using the proposed rules. For small scale instances, simple preprocessing can, on average, reduce 77% of the columns, while for medium and large scale instances the reduction drops to 58% and 20% respectively. The main reason for the drop in effectiveness is the quality of upper bounds obtained for each class of instance and the complexity of the analyzed instances. The results also reveal that the second probing algorithm is more effective than the first, and can reduce the number of columns even further. In total 85% of all columns are removed on average in the small scale instances, while for medium and large scale instances the number is 70% and 28% on average, respectively. The upper bounds computed initially (as warm start) are relatively tight and only two upper bounding models cannot be solved to optimality within 15 minutes. There is an average of 7% of optimality gap (see G_{UB} column for each instance size in Table 4). Model with small time limits also outperforms SWO heuristic for 8 instances of 10 large scale instances (see SWO $(Z_H)_{best}$ column in Table 5). These results show that the performance of GSPP formulations for small time limits is also strong. The results of GSPP (BL=1) without any probing (but including the two simple preprocessing methods) are presented in the last columns of Table 4. For this experiment there is no clear winner, but one can argue that the extra complexity involved in the probing algorithms does not pay off here.

Table 5 summarizes results for the time-invariant BACAP. For this problem, the known upper and lower bounds have been improved for all instances. The GSPP (BL=1) results outperform BL=2 and dynamic discretization results for all instances except #21, #23. For instance #23, BL=2 and dynamic discretization policies both present best upper bound, while for instance #21, dynamic discretization performs the best. The dynamic discretization policy still finds the optimum solutions of original problem (BL=1) for 14 out of 30 instances. We can conclude that finer discretization outperforms other discretization methods for most of the instances. The last column of Table 5 presents the results of squeaky wheel optimization (SWO) heuristic which is proposed by Meisel [17] for BACAP with time-invariant QC assignment (BL=1 version).

The performance of the GSPP (BL=1) can be compared with a modified version of the model by Meisel and Bierwirth [18] (presented in Section 3.2). Table 6 shows the results from this model. Upper and lower bounds, gaps, computational times, and the number of nodes in the B&B tree are presented. Results show that even for the small scale instances, there are two cases in which no integer solution was found. In this benchmark, only five instances are solved to optimality. The GSPP formulation solves all the instances to optimality in much shorter times. For medium and large scale instances, there are only two instances in which an upper bound is obtained and the lower bounds are significantly worse than those from the GSPP formulation.

Table 4: GSPP (BerthLength=1, Fixed QC)-Original Problem with Fixed QC number

	Computational Results of GSPP								Performance of Column Reduction				UB Generator		Results with probing disabled			
#	Z	LB	G_1	T_C	T_{OPT}	R_+^-	R_{LB}	#Nodes B&B	$\mid \Omega \mid$	$\mid \Omega_1 \mid$	$\mid \Omega_2 \mid$	$\mid \Omega_3 \mid$	UB T_{OPT}	Z_{UB}	Z	LB	G_2	T_{OPT}
1	89.0	89.0	0.0%	2	142	+	-	-	316278	102471	88598	62677	21	97.4	89.0	89.0	0.0%	90
2	56.2	56.2	0.0%	<1	<1	+	-	-	338438	19133	15785	687	27	57.4	56.2	56.2	0.0%	26
3	85.7	85.7	0.0%	1	186	+	-	-	252618	94083	84536	71287	28	88.9	85.7	85.7	0.0%	116
4	81.8	81.8	0.0%	1	160	+	-	-	390624	106970	99809	79138	47	94.4	81.8	81.8	0.0%	148
5	59.2	59.2	0.0%	1	31	+	-	-	275238	29887	24487	20471	15	61.6	59.2	59.2	0.0%	23
6	59.2	59.2	0.0%	<1	7	+	-	-	305485	27446	21672	11544	25	67.6	59.2	59.2	0.0%	54
7	75.2	75.2	0.0%	1	59	+	-	-	330639	58441	51442	26434	26	77.6	75.2	75.2	0.0%	161
8	61.4	61.4	0.0%	1	100	+	-	-	359703	61200	54227	48862	31	72.2	61.4	61.4	0.0%	117
9	79.0	79.0	0.0%	1	152	+	-	-	286683	91399	84533	66469	27	89.0	79.0	79.0	0.0%	179
10	101.0	101.0	0.0%	2	282	-	97.6	1480	359320	132706	126499	115110	48	106.1	101.0	101.0	0.0%	349
		G_{av}	0.0%						R_{av}	77%	12%	31%	G_{UB}	9%				
11	143.2	143.2	0.0%	6	418	+	-	-	496340	250999	231616	202530	63	151.4	143.2	143.2	0.0%	506
12	92.0	92.0	0.0%	1	84	+	-	-	523233	100586	88348	55483	36	95.0	92.0	92.0	0.0%	140
13	110.0	110.0	0.0%	2	143	+	-	-	483010	122017	110399	95635	25	112.8	110.0	110.0	0.0%	141
14	107.4	107.4	0.0%	3	161	+	-	-	467661	164949	154643	112752	50	117.2	107.4	107.4	0.0%	244
15	168.4	168.4	0.0%	5	621	+	-	-	477733	283078	273274	248677	76	181.8	168.4	168.4	0.0%	753
16	121.6	121.6	0.0%	3	169	+	-	-	532026	181721	167068	102007	71	122.8	121.6	121.6	0.0%	376
17	109.4	109.4	0.0%	3	119	+	-	-	467085	142579	130493	92707	45	114.0	109.4	109.4	0.0%	134
18	135.0	135.0	0.0%	8	556	-	131.9	158	508500	251673	240444	233079	70	150.2	135.0	135.0	0.0%	465
19	176.2	176.2	0.0%	15	748	-	174.3	3259	506300	312026	302952	272114	77	189.4	176.2	176.2	0.0%	890
20	139.8	139.8	0.0%	11	475	+	-	-	487945	238218	224054	163891	63	145.8	139.8	139.8	0.0%	451
		G_{av}	0.0%						R_{av}	58%	7%	20%	G_{UB}	6%				
21	247.0	233.1	5.6%	11	36000	-	220.9	117418	768104	614655	605174	577530	290	253.2	246.8	242.8	1.6%	36000
22	178.4	178.4	0.0%	10	532	+	-	-	611023	403976	386376	295991	83	189.0	178.4	178.4	0.0%	821
23	269.0	247.8	7.7%	21	36000	-	242.0	34137	720952	616879	608858	589421	417	281.6	268.0	247.4	7.7%	36000
24	307.0	283.8	7.6%	32	36000	-	272.7	36504	657415	634437	625994	601113	900	326.8	309.8	283.6	8.4%	36000
25	164.6	164.6	0.0%	7	1012	-	158.8	4953	535300	358354	335868	309849	88	176.4	164.6	164.6	0.0%	1277
26	258.2	258.2	0.0%	23	31523	-	236.4	118708	675819	596031	586485	552397	879	278.9	258.2	258.2	0.0%	33669
27	205.4	205.4	0.0%	10	1206	-	198.8	2909	615092	474731	459672	427398	85	220.2	205.4	205.4	0.0%	1523
28	300.1	267.5	10.9%	35	36000	-	256.4	27200	698783	631227	627241	612621	900	320.2	299.5	270.6	9.7%	36000
29	227.6	227.6	0.0%	26	8077	-	212.1	34227	673090	501858	493100	454074	132	241.4	227.6	227.6	0.0%	3629
30	210.1	210.1	0.0%	31	20966	-	196.0	108359	678072	523970	508499	463573	559	235.6	210.1	210.1	0.0%	25400
		G_{av}	3.2%						R_{av}	20%	2%	7%	G_{UB}	7%				

$G_1 = [\frac{Z-LB}{Z}]$, $G_2 = [\frac{Z_{w.Prober}-LB_{w.Prober}}{Z_{w.Prober}}]$, $(R_{av})_i = [\frac{|\Omega|-|\Omega_i|}{|\Omega|}]$, $G_{UB} = [\frac{Z_{UB}-Z_{w.Prober}}{Z_{UB}}]$, $G_{av} = [\frac{\sum_{i\in n} G_1}{n}]$.

Table 5: Results of GSPP, SWO heuristic-BACAP-TI (Meisel [17])

		GSPP (BL=1)				GSPP (BL=2)				GSPP (Dynamic)			SWO
N	#	Z	LB	G		Z	LB	G		Z	LB	G	$(Z_H)_{best}$
20	1	89.0	89.0	0.0%		92.8	92.8	0.0%		91.4	91.4	0.0%	89.0
	2	56.2	56.2	0.0%		56.2	56.2	0.0%		56.2	56.2	0.0%	56.2
	3	85.7	85.7	0.0%		87.7	87.7	0.0%		85.7	85.7	0.0%	89.8
	4	81.8	81.8	0.0%		94.4	94.4	0.0%		81.8	81.8	0.0%	81.8
	5	59.2	59.2	0.0%		60.4	60.4	0.0%		60.4	60.4	0.0%	59.2
	6	59.2	59.2	0.0%		62.8	62.8	0.0%		59.2	59.2	0.0%	59.2
	7	75.2	75.2	0.0%		77.0	77.0	0.0%		77.0	77.0	0.0%	75.8
	8	61.4	61.4	0.0%		68.4	68.4	0.0%		61.4	61.4	0.0%	61.4
	9	79.0	79.0	0.0%		79.0	79.0	0.0%		79.0	79.0	0.0%	79.0
	10	101.0	101.0	0.0%		102.5	102.5	0.0%		102.0	102.0	0.0%	105.0
			G_{av}	0.0%			G_{av}	0.0%			G_{av}	0.0%	
			$(T_{opt})_{av}$	126			$(T_{opt})_{av}$	36			$(T_{opt})_{av}$	76	
30	11	143.2	143.2	0.0%		143.2	143.2	0.0%		143.2	143.2	0.0%	148.0
	12	92.0	92.0	0.0%		92.0	92.0	0.0%		92.0	92.0	0.0%	93.4
	13	110.0	110.0	0.0%		110.0	110.0	0.0%		110.0	110.0	0.0%	111.2
	14	107.4	107.4	0.0%		112.4	112.4	0.0%		107.6	107.6	0.0%	115.8
	15	168.4	168.4	0.0%		170.8	170.8	0.0%		170.8	170.8	0.0%	175.8
	16	121.6	121.6	0.0%		121.8	121.8	0.0%		121.6	121.6	0.0%	126.6
	17	109.4	109.4	0.0%		109.4	109.4	0.0%		109.4	109.4	0.0%	114.6
	18	135.0	135.0	0.0%		143.0	143.0	0.0%		136.2	136.2	0.0%	144.4
	19	176.2	176.2	0.0%		179.8	179.8	0.0%		176.2	176.2	0.0%	180.8
	20	139.8	139.8	0.0%		141.0	141.0	0.0%		139.8	139.8	0.0%	139.8
			G_{av}	0.0%			G_{av}	0.0%			G_{av}	0.0%	
			$(T_{opt})_{av}$	410			$(T_{opt})_{av}$	107			$(T_{opt})_{av}$	234	
40	21	247.0	233.1	5.6%		250.6	250.6	0.0%		246.8	246.8	0.0%	293.2
	22	178.4	178.4	0.0%		185.4	185.4	0.0%		180.6	180.6	0.0%	193.8
	23	269.0	247.8	7.7%		268.6	268.6	0.0%		268.6	264.2	1.7%	331.4
	24	307.0	283.8	7.6%		310.6	303.8	2.2%		307.0	292.5	4.7%	366.0
	25	164.6	164.6	0.0%		168.2	168.2	0.0%		165.8	165.8	0.0%	171.6
	26	258.2	258.2	0.0%		265.2	265.2	0.0%		258.2	258.2	0.0%	278.4
	27	205.4	205.4	0.0%		216.8	216.8	0.0%		206.6	206.6	0.0%	235.8
	28	300.1	267.5	10.9%		314.8	282.6	10.2%		313.1	272.1	13.1%	412.4
	29	227.6	227.6	0.0%		235.2	235.2	0.0%		229.4	229.4	0.0%	255
	30	210.1	210.1	0.0%		224.4	224.4	0.0%		217.0	217.0	0.0%	284.2
			G_{av}	3.2%			G_{av}	1.2%			G_{av}	1.9%	
			$(T_{opt})_{av}$	21032			$(T_{opt})_{av}$	11565			$(T_{opt})_{av}$	14160	

Table 6: Reformulation of compact model-BACAP-TI

#	Z	LB	G	T_{OPT}	#Nodes		#	Z	LB	G	#Nodes		#	Z	LB	#Nodes
1*	89.0	86.2	3.1%	*	2769450		11*	150.4	108.3	27.9%	1557183		21*	X	116.4	762709
2	56.2	56.2	0.0%	7397	383896		12*	X	74.1	-	1205386		22*	X	97.5	438579
3*	X	67.6	-	*	2605595		13*	X	88.3	-	1408812		23*	X	142.5	745119
4*	X	68.7	-	*	2166910		14*	X	85.3	-	1284386		24*	X	139.4	415411
5	59.2	59.2	0.0%	4151	466141		15*	X	97.3	-	1427223		25*	X	115.9	824894
6	59.2	59.2	0.0%	105	5428		16*	X	96.2	-	1500854		26*	X	102.4	433046
7*	75.2	75.1	0.16%	*	1748332		17*	X	87.9	-	1311279		27*	X	133.4	947127
8	61.4	61.4	0.0%	12057	876111		18*	135.7	106.5	21.5%	1694085		28*	X	140.8	721192
9	79.0	79.0	0.0%	4464	385267		19*	X	118.0	-	1019351		29*	X	123.0	792108
10*	101.0	90.5	10.4%	*	4744212		20*	X	105.1	-	1135501		30*	X	128.6	905548

$G = [\frac{Z-LB}{Z}]$, * represents that 10h time-limit has been reached

Finally, the performance of dynamic discretization which shows promising results is evaluated in detail. Dynamic discretization models have less columns than the BL=1 case, but more than the BL=2. Since discretization is one in 5

units around the desired berthing position, and two for the rest, the number of columns is closer to the GSPP (BL=2). On average, the number of columns is 44% less than the BL=1 case. For small and medium instances, the computational time needed to reach optimality is reduced by 50%, and 54% compared to BL=1 case ($T_{GAP} = \frac{T_{OPT.Dynamic} - T_{OPT.BL=1}}{T_{OPT.BL=1}}$). The results show that, with dynamic discretization, optimal solutions can be obtained for all but 3 instances (#23, #24, #28). It is observed that for instances (#21, #23), which are not solved to optimality in the BL=1, dynamic discretization obtains a lower objective value in 10 hours of computational time.

Figure 3 illustrates the gap between each discretization policy and the best known solution for each instance. Apart from instance (#21, #23), BL=1 discretization presents the best upper bounds for each instance. Dynamic discretization performs better than BL=2 case in all instances (in some instance they found identical solutions).

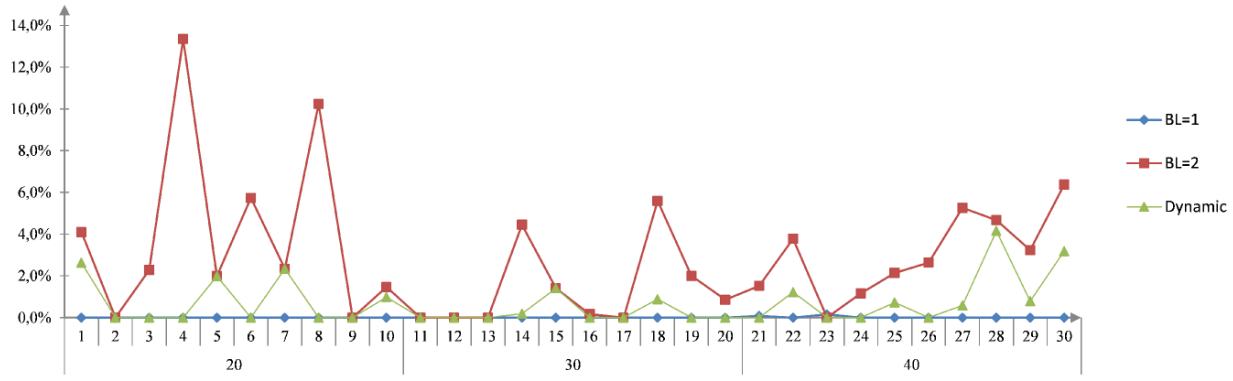

Figure 3: Gap (%) from best known solution for BL=1, BL=2 and dynamic discretized BACAP- Time-Invariant QC policy

5.3.2. Time-Variant GSPP results

Tables 7 and 8 present the results of the time-variant version of the GSPP formulation (see, Section 4.2). As before, we first present results for a berth length of 1 unit. Afterwards, we present summary of results for different discretization policies and solution approaches.

The results from the BL=1 case are reported in Table 7. This problem corresponds to the one solved by Meisel and Bierwirth [18]. The results in Table 7 show that only one of the small instances cannot be solved to optimality. For medium instances, the average gap is less than 2%, while for large instances it is around 15%. It should be noted that the standard deviation of the optimality gap for large instances is high.

The column reduction techniques remove a larger fraction of the columns compared to the time-invariant case, but the number of surviving columns has nevertheless increased by a factor of 2.8 on average. Overall, all four reduction tools have deleted 90% of the columns in the small instances, while 72% and 27% are reduced for medium and large scale instances, respectively. The computational time to generate the columns remains acceptable for small and medium scale instances, for large scale instances the average time is 10 minutes.

Compared to the time-invariant case the complexity of the model has increased and it becomes reasonable to see that the quality of the warms start solutions have decreased. For several large scale instances, the model faces memory issues. Six of them ran out of memory and were rerun using only one thread (which reduces memory usage).

Table 7: GSPP (BerthLength=1, Variable QC assignment)

	Computational results of GSPP								Performance of column reduction				UB Gen.	
#	Z	LB	G_1	T_C	T_{OPT}	R_+^-	R_{LB}	#Nodes B&B	$\mid \Omega \mid$	$\mid \Omega_1 \mid$	$\mid \Omega_2 \mid$	$\mid \Omega_3 \mid$	UB T_{OPT}	Z_{UB}
1	84.1	84.1	0.0%	20	321	-	83.3	2887	1234646	317414	261017	159035	142	89.0
2	53.9	53.9	0.0%	1	2	+	-	-	1086872	68846	57230	4265	1	56.2
3	76.3	76.3	0.0%	4	1096	-	75.0	1081145	936469	321403	279470	242882	186	85.7
4	76.2	76.2	0.0%	3	717	-	74.0	82907	1366755	255422	233926	153491	160	81.8
5	56.8	56.8	0.0%	<1	56	+	-	-	882054	87858	71329	54527	31	59.2
6	57.6	57.6	0.0%	<1	1	+	-	-	1103729	23934	16177	1411	7	59.2
7	68.0	68.0	0.0%	7	203	-	67.1	2614	1197949	210198	184265	85013	59	75.2
8	56.1	56.1	0.0%	1	87	-	55.9	45	1193809	111140	89978	71392	100	61.4
9	75.1	75.1	0.0%	5	924	-	73.2	72287	954820	240732	213842	140247	152	79.0
10	89.3	88.4	1.1%	19	36000	-	83.1	813843	1281315	464115	440292	391410	282	101.0
		G_{av}	0.1%						R_{av}	82%	16%	43%	G_{UB}	8%
11	139.8	139.1	0.5%	58	36000	-	133.8	219068	1744848	816725	744802	616533	418	143.2
12	81.8	81.8	0.0%	3	337	+	-	-	1607596	341504	302519	189863	84	92.0
13	102.4	102.4	0.0%	11	1122	-	100.5	10488	1727182	429088	381602	334247	143	110.0
14	99.1	99.1	0.0%	23	12960	-	97.4	5798205	1525995	475467	440336	281220	161	107.4
15	154.2	145.1	5.9%	27	36000	-	136.8	30374	1761414	995156	951700	830382	621	168.4
16	115.4	111.6	3.3%	29	36000	-	107.8	330293	1796509	670045	618100	381185	169	121.6
17	102.6	102.6	0.0%	20	1416	-	100.7	41326	1696500	474650	429690	293693	119	109.4
18	121.9	120.0	1.6%	69	36000	-	117.4	804373	1709858	726816	677637	634660	556	135.0
19	165.2	161.6	2.2%	349	36000	-	158.8	56899	2004197	1195939	1153256	1001574	748	176.2
20	132.5	129.6	2.2%	243	36000	-	128.6	144075	1801078	812503	750895	521506	475	139.8
		G_{av}	1.6%						R_{av}	61%	8%	24%	G_{UB}	8%
21†	223.1	178.3	20.1%	51	36000	-	178.0	326	2513805	2056806	2023788	1928308	1200	250.6
22	171.0	166.1	2.9%	445	36000	-	163.1	37143	2010843	1370334	1310042	963164	239	185.4
23†	264.4	196.7	25.6%	335	36000	-	196.4	27	2364293	2014422	1983473	1923115	1200	269.2
24‡	302.2	245.4	18.8%	846	36000	-	239.4	22	2258599	2176590	2144639	2065263	1200	320.4
25	148.1	141.2	4.7%	52	36000	-	139.9	10063	1690236	1117012	1041771	943454	195	168.2
26†	245.0	208.8	14.8%	1327	36000	-	208.0	300	2269042	2005944	1965226	1831379	1200	265.2
27	184.9	176.0	4.8%	113	36000	-	173.9	5507	2017218	1612962	1560262	1460709	423	216.8
28‡	315.2	215.3	31.7%	1499	36000	-	215.3	15	2543836	2340412	2327678	2277105	1200	320.2
29	226.7	186.0	18.0%	433	36000	-	185.5	974	2163279	1669144	1643406	1495597	350	235.2
30†	187.8	175.0	6.8%	477	36000	-	173.9	4103	2441545	1875349	1811354	1614804	1200	226.7
		G_{av}	14.8%						R_{av}	19%	3%	8%	G_{UB}	12%

† represents that CPLEX solver has faced a memory overflow while the instance was running using 5 threads. Hence, † instances are rerun with only one thread. ‡ represents instances that run with one thread, and again faced a memory problem. The results are reported on the time when CPLEX ran out of memory.

We can conclude that the GSPP model can solve the problem considered in Meisel and Bierwirth [18] to optimality, or near optimality, for instances with 20 or 30 vessels. For larger instances the performance is more erratic and only a subset of those instances can be solved reasonably well. In the following the performance of the time-variant GSPP model is compared with the compact model proposed by Meisel and Bierwirth [18]. Table 8 summarizes the results. In addition to the upper and lower bounds of the compact model, the best results from the heuristic procedures described in Meisel and Bierwirth [18] are presented. Additionally, the results from the time-variant GSPP model and the Meisel and Bierwirth [18] model with warm-start solutions are presented in Table 8. Results show that there are only four instances which are solved to optimality by the compact model, while 13 instances are solved to optimality by the BL=1 discretized GSPP model. The compact model performs very poorly on the medium and large scale instances in terms of producing upper bounds, and the lower bounds are consistently outperformed by the GSPP model. When warm-starts are imposed on Meisel and Bierwirth [18] model, eight instances are solved to optimality, and four instances resulted in a better upper bound compared to the GSPP models. We also observe that the upper bounds produced by all versions of the GSPP models often improve upon the best known upper bounds produced by the state-of-the-art heuristics. We hope that the improved upper and lower bounds will be helpful in evaluating future heuristics and exact methods for the problem.

Table 8 also helps to compare performance of different discretization methods. Since BL=1 discretization solves many of small and medium scale instances to optimality, BL=1 outperforms dynamic and BL=2 discretization for all such

Table 8: Results of Model, heuristic results-BACAP-TV (Meisel and Bierwirth [18]), GSPP results

		Compact Model			Compact Model with warmstart			GSPP (BL=1)			GSPP (Dynamic)			GSPP (BL=2)			Heuristic
N	#	Z	LB	G	Z	LB	G	Z	LB	G	Z	LB	G	Z	LB	G	$(Z_H)_{best}$
20	1	84.1	84.0	0.1%	84.1	84.1	0.0%	84.1	84.1	0.0%	86.1	86.1	0.0%	87.3	87.3	0.0%	85.1
	2	53.9	53.9	0.0%	53.9	53.9	0.0%	53.9	53.9	0.0%	53.9	53.9	0.0%	54.5	54.5	0.0%	53.9
	3	77.4	75.2	2.9%	76.3	76.1	0.3%	76.3	76.3	0.0%	77.4	77.4	0.0%	77.6	77.6	0.0%	77.4
	4	76.2	75.8	0.5%	76.2	75.9	0.4%	76.2	76.2	0.0%	77.9	77.9	0.0%	78.5	78.5	0.0%	77.9
	5	56.8	56.8	0.0%	56.8	56.8	0.0%	56.8	56.8	0.0%	58.1	58.1	0.0%	58.2	58.2	0.0%	56.8
	6	57.6	57.6	0.0%	57.6	57.6	0.0%	57.6	57.6	0.0%	57.6	57.6	0.0%	60.8	60.8	0.0%	57.6
	7	68.0	67.5	0.7%	68.0	68.0	0.0%	68.0	68.0	0.0%	69.3	69.3	0.0%	69.5	69.5	0.0%	68.9
	8	56.1	56.1	0.0%	56.1	56.1	0.0%	56.1	56.1	0.0%	57.2	57.2	0.0%	60.5	60.5	0.0%	56.1
	9	75.1	75.0	0.1%	75.1	75.1	0.0%	75.1	75.1	0.0%	75.3	75.3	0.0%	75.4	75.4	0.0%	75.5
	10	90.9	88.2	3.0%	89.3	88.9	0.4%	89.3	88.4	1.1%	89.4	88.4	1.1%	89.8	89.8	0.0%	93.0
			G_{av}	0.7%		G_{av}	0.1%		G_{av}	0.1%		G_{av}	0.1%		G_{av}	0.0%	
			$(T_{opt})_{av}$	-		$(T_{opt})_{av}$	11643		$(T_{opt})_{av}$	3939		$(T_{opt})_{av}$	5983		$(T_{opt})_{av}$	2500	
30	11	X	137.7	-	140.4	130.0	7.4%	139.8	139.1	0.5%	140.0	139.7	0.2%	140.5	140.5	0.0%	147.8
	12	81.8	81.4	0.4%	81.8	81.8	0.0%	81.8	81.8	0.0%	82.5	82.5	0.0%	82.7	82.7	0.0%	82.5
	13	104.9	100.9	3.9%	102.4	101.8	0.6%	102.4	102.4	0.0%	103.5	103.1	0.4%	104.8	103.7	1.1%	104.5
	14	X	96.8	-	99.1	98.7	0.4%	99.1	99.1	0.0%	102.0	101.8	0.2%	105.3	105.3	0.0%	105.8
	15	X	136.9	-	150.4	132.1	12.1%	154.2	145.1	5.9%	154.7	150.7	2.6%	154.9	152.9	1.3%	157.4
	16	X	106.2	-	113.8	109.1	4.1%	115.4	111.6	3.3%	113.8	112.8	0.9%	115.4	114.2	1.0%	118.5
	17	X	99.6	-	102.6	101.9	0.6%	102.6	102.6	0.0%	102.9	102.7	0.2%	103.2	103.2	0.0%	104.2
	18	X	117.8	-	121.9	118.8	2.5%	121.9	120.0	1.6%	122.8	120.9	1.5%	128.0	126.0	1.6%	125.5
	19	X	156.4	-	165.2	146.1	11.5%	165.2	161.6	2.2%	166.4	165.1	0.8%	168.2	168.2	0.0%	173.8
	20	X	125.6	-	134.5	120.8	10.2%	132.5	129.6	2.2%	132.8	131.8	0.8%	133.1	132.7	0.3%	135.2
			G_{av}	2.2%		G_{av}	4.9%		G_{av}	1.5%		G_{av}	0.8%		G_{av}	0.5%	
			$(T_{opt})_{av}$	-		$(T_{opt})_{av}$	33950		$(T_{opt})_{av}$	23183		$(T_{opt})_{av}$	32418		$(T_{opt})_{av}$	21363	
40	21	X	165.7	-	206.3	130.4	36.8%	223.1	178.3	20.1%	209.0	180.8	13.5%	205.0	184.6	9.9%	215.0
	22	X	159.6	-	170.9	145.4	14.9%	171.0	166.1	2.9%	171.2	168.7	1.5%	175.3	173.0	1.3%	178.8
	23	X	185.0	-	257.7	135.1	47.6%	264.4	196.7	25.6%	264.5	197.3	25.4%	257.7	199.6	22.5%	264.3
	24	X	224.1	-	288.9	184.1	36.3%	302.2	245.4	18.8%	303.8	246.2	19.0%	292.6	251.5	14.1%	326.6
	25	X	133.3	-	148.9	125.2	15.9%	148.1	141.2	4.7%	148.3	144.8	2.4%	151.8	149.5	1.5%	155.1
	26	X	201.3	-	238.7	152.1	36.3%	245.0	208.8	14.8%	245.2	211.4	13.8%	245.9	221.1	10.1%	259.6
	27	X	172.2	-	183.9	148.7	19.1%	184.9	176.0	4.8%	183.9	178.6	2.9%	191.4	186.3	2.7%	200.8
	28	X	211.7	-	273.0	147.2	46.1%	315.2	215.3	31.7%	294.7	218.9	25.7%	256.5	220.1	14.2%	286.2
	29	X	180.3	-	214.1	146.4	31.6%	226.7	186.0	18.0%	211.9	187.9	11.3%	210.0	196.9	6.2%	219.4
	30	X	170.1	-	189.3	154.6	18.3%	187.8	175.0	6.8%	187.5	180.7	3.6%	193.4	186.4	3.6%	240.9
			G_{av}	-		G_{av}	30.2%		G_{av}	14.8%		G_{av}	11.9%		G_{av}	8.6%	
			$(T_{opt})_{av}$	-		$(T_{opt})_{av}$	36000		$(T_{opt})_{av}$	36000		$(T_{opt})_{av}$	36000		$(T_{opt})_{av}$	36000	

$(Z_H)_{best}$: The best solution of an instance for three heuristics (SWO. TS, FCFS with local refinements)

instances except instance #16, for which dynamic discretization presents the best upper bound. For small and medium scale instances, dynamic discretization continuously outperforms BL=2. For large scale instances, comparison is more interesting. BL=1 discretization performs better for three out of ten instances. While, BL=2 outperforms the other two methods in five instances, and dynamic discretization does the same with two instances.

5.3.3. Results for the berth allocation problem

The GSPP model, tested until now, can also solve the BAP by simply fixing the QC assignment decisions. This would result in a model similar to the one presented by Buhrkal et al. [4]. The latter, however, has only been tested for the discrete variant of the problem, where each berth holds exactly one vessel. Here we show how such models perform on the continuous variant. This problem is a special case of the BACAP.

We changed the same set of instances to include the number of QCs assigned (and consequently the processing time) as a parameter. The required change in the benchmark is achieved by replacing r_i^{min}, r_i^{max} in each data set with $\overline{r_i} = \left\lceil \frac{r_i^{min}+r_i^{max}}{2} \right\rceil$. Hence, the generation of columns will be based on the single value of the $\overline{r_i}$ parameter. The time-invariant version of GSPP without the knapsack constraints (constraint (26)) will be used for this experiment. The rest of the parameters are kept the same. We use the instances from Meisel and Bierwirth [18] and solely test the BL=1 discretization. Column reduction techniques were not applied and no upper bounds were computed a-priori.

The results are summarized in Table 9. Each row corresponds to 10 instances, the first column reports the instance size, while the next shows the average optimality gap (all instances were solved to optimality). The next column shows the average time needed to generate the model, then follows the time for solving the IP model and the last column shows the average number of columns generated. For the two last columns the number in square brackets indicates the standard deviation (σ_N) and coefficient of variation (σ_N/μ).

Table 9: Results for the BAP without quay crane decisions

N	G_{av}	T_c	T_{OPT}	$\mid \Omega \mid$
20	0.00%	3	131 [40,0.30]	141391 [17472,0.12]
30	0.00%	5	340 [143,0.42]	215967 [9128,0.04]
40	0.00%	6	893 [541,0.60]	286668 [25173,0.08]

The results show that the GSPP model is able to handle the fine discretization very well and the application of the model to the standard BAP can be taken further than what was done by Buhrkal et al. [4]. It seems likely that even larger instances could be solved within a couple of hours, but we have not tested this. We also conducted tests with the BL=2 discretization and the dynamic discretization. For the 40 ship instances the average solution costs were 2.8% and 1.71% higher than those from the BL=1 discretized model, respectively. Meanwhile the average running time was 209 seconds for BL=2 models and 606 seconds for dynamic discretized models. Based on the results, we recommend using the BL=1 model for instances of this size or smaller.

6. Conclusions and suggestions for future work

In this paper, we have proposed novel GSPP formulations for the BACAP considering both time-variant and time-invariant QC assignment policies. The proposed models solve the problem introduced in Meisel and Bierwirth [18]. Computational results show that the performances of both the time-variant and time-invariant GSPP formulations are strong with respect to both upper and lower bounds. In particular, the GSPP formulation can provide optimal solutions in relatively short computation times for the small and medium sized instances. For these instance sizes, all instances could be solved to optimality for the time-invariant case, while 13 out of 20 instances could be solved for the time-variant case. For large scale instances, the objective value and lower bounds have been improved. We believe that the improved bounds would be useful in the evaluation of new heuristics to solve such instances. Note that both upper and lower bounds have been improved compared to the state-of-the-art results for all 60 instances when the results of compact model with warm start is also taken into account, and for 56 of 60 instances otherwise. This paper also discusses the effects of time-variant and time-invariant QC assignment policy for terminals. We show that there is an additional cost of time-invariant QC policy and we quantify this difference, although for artificial instances.

The GSPP model has also been used to solve classical berth allocation problems with a fine discretization of the berthing space, and the results show that the model is very effective.

The presented set partitioning models contain many variables and therefore several variable reduction methods are proposed and evaluated. These novel column reduction techniques for the BACAP can reduce the number of columns by up to 90% for some benchmarks. By using the proposed reduction techniques, in most cases, we create models that can be handled in the memory available in current computers, however this is not always the case. We believe that the reduction techniques can be generalized to other variants of the berth allocation problem.

A convenient property of the proposed solution method is that most of the work is done by a black-box IP solver (in this case CPLEX). This means that the solution approach will automatically benefit from new developments in solver techniques and will also benefit from future hardware improvements that are supported by the underlying IP solver (for example a far more massive parallelism).

Future research could be directed towards including more constraints in the model, if that is deemed necessary to apply the model in a particular port or it could be directed at designing improved solution methods. Since a major limitation of the proposed model is the rapid growth in the number of variables with increase in problem size, a natural extension of the current work is to attempt to generate variables dynamically using delayed column generation and solve the model using a branch-and-price algorithm.

7. Acknowledgments

The authors would like to thank Frank Meisel (Kiel-University) for contributing benchmark instances of studied problem, and for his helpful comments. This project is supported by the Danish Maritime Cluster under DTU research projects 35307.

AppendixA. Proof of Theorem 1

Assume that a better solution existed (resulting in fewer QC hours used). Let h_r be the number of hours we use r QCs in this solution, and h be the duration of the port stay ($d(j)$ for given column j). We must have:

$$\sum_{r\in\{r_{min},\ldots,r_{max}\}} h_r = h \tag{A.1}$$

in order for the preemption constraint to be satisfied.

- (a) If $h_r = 0$ for all $r \in \{r_{min}, \ldots, r_{max}\}\backslash\{q, q{+}1\}$ then we cannot do better than the solution computed in Algorithm (1) since here we computed all possible combinations by using q and $q+1$ QCs.
- (b) Therefore, let us first analyze the situation where $h_r = 0$ for all $r < q$ and $h_{\bar{r}} > 0$ for some $\bar{r} > q+1$. If $h_q = 0$ then this solution is clearly worse than the one computed by the algorithm because of constraint (A.1). Thus we must have $h_q > 0$ and we can make a more balanced solution that uses the same amount of QC hours by incrementing h_{q+1} and $h_{\bar{r}-1}$ by one (if $q+1 = \bar{r}-1$ then we increase h_{q+1} by 2) and decrementing h_q and $h_{\bar{r}}$ by one. We continue doing so until either $h_q = 0$, showing that the starting solution was not better than the one computed by the algorithm or $h_q > 0$ and $h_{\bar{r}} = 0$ for all $\bar{r} > q+1$. Now we are back at case (a) and we see that the starting solution could not have used fewer QC hours than the one computed by the algorithm.
- (c) Now let us analyze the situation where $h_r > 0$ for $r < q$. In that case we must have $h_{\bar{r}} > 0$ for some $\bar{r} > q$. Otherwise we would have selected a lower q in the initial checks. We construct a more balanced solution that use the same number of QC hours by increasing h_{r+1} and $h_{\bar{r}-1}$ by one (if $r+1 = \bar{r}-1$ then this should be interpreted as increasing h_{r+1} by 2) and decreasing h_r and $h_{\bar{r}}$ by one. By continuing to do so we either get to situation (a) or (b) and we see that the starting solution could not have used fewer QC hours than the one computed by the algorithm. □

AppendixB. References

[1] Christian Bierwirth and Frank Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202(3):615–627, May 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.05.031. URL http://linkinghub.elsevier.com/retrieve/pii/S0377221709003579.

[2] Christian Bierwirth and Frank Meisel. A follow-up survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 000:1–15, December 2014. ISSN 03772217. doi: 10.1016/j.ejor.2014.12.030. URL http://linkinghub.elsevier.com/retrieve/pii/S0377221714010480.

[3] J Blazewicz, T C E Cheng, M Machowiak, and C Oguz. Berth and quay crane allocation: a moldable task scheduling model. *Journal of the Operational Research Society*, 62(7):1189–1197, June 2010. ISSN 0160-5682. doi: 10.1057/jors.2010.54. URL http://www.palgrave-journals.com/doifinder/10.1057/jors.2010.54.

[4] Katja Buhrkal, Sara Zuglian, Stefan Ropke, Jesper Larsen, and Richard Lusby. Models for the discrete berth allocation problem: A computational comparison. *Transportation Research Part E: Logistics and Transportation Review*, 47 (4):461–473, July 2011. ISSN 13665545. doi: 10.1016/j.tre.2010.11.016. URL http://linkinghub.elsevier.com/retrieve/pii/S1366554510001201.

[5] Daofang Chang, Zuhua Jiang, Wei Yan, and Junliang He. Integrating berth allocation and quay crane assignments. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):975–990, November 2010. ISSN 13665545. doi: 10.1016/j.tre.2010.05.008. URL http://linkinghub.elsevier.com/retrieve/pii/S1366554510000608.

[6] Jiang Hang Chen, Der-Horng Lee, and Jin Xin Cao. A combinatorial benders cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):266–275, January 2012. ISSN 13665545. doi: 10.1016/j.tre.2011.06.004. URL http://linkinghub.elsevier.com/retrieve/pii/S1366554511000834.

[7] Clement Gram Christensen and Cecilie Terese Holst. *Berth Allocation in Container Terminals (In Danish)*. PhD thesis, Technical University of Denmark, 2008. URL http://etd.dtu.dk/thesis/219860/.

[8] Jean-François Cordeau, Gilbert Laporte, Pasquale Legato, and Luigi Moccia. Models and Tabu Search Heuristics for the Berth-Allocation Problem. *Transportation Science*, 39(4):526–538, November 2005. ISSN 0041-1655. doi: 10.1287/trsc.1050.0120. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1050.0120.

[9] Giovanni Giallombardo, Luigi Moccia, Matteo Salani, and Ilaria Vacca. Modeling and solving the Tactical Berth Allocation Problem. *Transportation Research Part B: Methodological*, 44(2):232–245, February 2010. ISSN 01912615. doi: 10.1016/j.trb.2009.07.003. URL http://linkinghub.elsevier.com/retrieve/pii/S0191261509000824.

[10] G. Harche, f, and Thompson. THE COLUMN SUBTRACTION ALGORITHM : AN EXACT METHOD FOR SOLVING WEIGHTED SET COVERING , PACKING AND PARTITIONING PROBLEMS. *Computers & Operations Research*, 21(6):689–705, 1994.

[11] Akio Imai, Etsuko Nishimura, and Stratos Papadimitriou. The dynamic berth allocation problem for a container port. *Transportation Research Part B: Methodological*, 35(4):401–417, May 2001. ISSN 01912615. doi: 10.1016/S0191-2615(99)00057-0. URL http://linkinghub.elsevier.com/retrieve/pii/S0191261599000570.

[12] Akio Imai, Xin Sun, Etsuko Nishimura, and Stratos Papadimitriou. Berth allocation in a container port: using a continuous location space approach. *Transportation Research Part B: Methodological*, 39(3):199–221, March 2005. ISSN 01912615. doi: 10.1016/j.trb.2004.04.004. URL http://linkinghub.elsevier.com/retrieve/pii/S0191261504000505.

[13] Akio Imai, Hsieh Chia Chen, Etsuko Nishimura, and Stratos Papadimitriou. The simultaneous berth and quay crane allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 44(5):900–920, September 2008. ISSN 13665545. doi: 10.1016/j.tre.2007.03.003. URL http://linkinghub.elsevier.com/retrieve/pii/S1366554507000555.

[14] Kap Hwan Kim and Kyung Chan Moon. Berth scheduling by simulated annealing. *Transportation Research Part B: Methodological*, 37(6):541–560, July 2003. ISSN 01912615. doi: 10.1016/S0191-2615(02)00027-9. URL http://linkinghub.elsevier.com/retrieve/pii/S0191261502000279.

[15] Chengji Liang, Youfang Huang, and Yang Yang. A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning. *Computers & Industrial Engineering*, 56(3):1021–1028, April 2009. ISSN 03608352. doi: 10.1016/j.cie.2008.09.024. URL http://linkinghub.elsevier.com/retrieve/pii/S0360835208002039.

[16] Jiyin Liu, Yat-wah Wan, and Lei Wang. Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures. *Naval Research Logistics*, 53(1):60–74, February 2006. ISSN 0894-069X. doi: 10.1002/nav.20108. URL http://doi.wiley.com/10.1002/nav.20108.

[17] Frank Meisel. Seaside Operations Planning in Container Terminals. 2009. doi: 10.1007/978-3-7908-2191-8. URL http://link.springer.com/10.1007/978-3-7908-2191-8.

[18] Frank Meisel and Christian Bierwirth. Heuristics for the integration of crane productivity in the berth allocation problem. *Transportation Research Part E: Logistics and Transportation Review*, 45(1):196–209, January 2009. ISSN 13665545. doi: 10.1016/j.tre.2008.03.001. URL http://linkinghub.elsevier.com/retrieve/pii/S1366554508000768.

[19] Frank Meisel and Christian Bierwirth. A Framework for Integrated Berth Allocation and Crane Operations Planning in Seaport Container Terminals. *Transportation Science*, 47(2):131–147, May 2013. ISSN 0041-1655. doi: 10.1287/trsc.1120.0419. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0419.

[20] Young-Man Park and Kap Hwan Kim. A scheduling method for Berth and Quay cranes. *OR Spectrum*, 25(1):1–23, February 2003. ISSN 0171-6468. doi: 10.1007/s00291-002-0109-z. URL http://link.springer.com/10.1007/s00291-002-0109-z.

[21] Birger Raa, Wout Dullaert, and Rowan Van Schaeren. An enriched model for the integrated berth allocation and quay crane assignment problem. *Expert Systems with Applications*, 38(11):14136–14147, May 2011. ISSN 09574174. doi: 10.1016/j.eswa.2011.04.224. URL http://linkinghub.elsevier.com/retrieve/pii/S0957417411007512.

[22] Tomáš Robenek, Nitish Umang, Michel Bierlaire, and Stefan Ropke. A branch-and-price algorithm to solve the integrated berth allocation and yard assignment problem in bulk ports. *European Journal of Operational Research*, 235(2):399–411, 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.08.015. URL http://linkinghub.elsevier.com/retrieve/pii/S0377221713006802.

[23] Yousra Saadaoui, Nitish Umang, and Emma Frejinger. A Column Generation Framework for Berth Scheduling at Port Terminals. Technical report, 2015. URL https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2015-15.pdf.

[24] Robert Stahlbock and Stefan Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, October 2007. ISSN 0171-6468. doi: 10.1007/s00291-007-0100-9. URL http://link.springer.com/10.1007/s00291-007-0100-9.

[25] Yavuz B. Turkogullari, Z. Caner Taskin, Necati Aras, and I. Kuban Altinel. Optimal berth allocation and time-invariant quay crane assignment in container terminals. *European Journal of Operational Research*, 235(1):88–101, May 2014. ISSN 03772217. doi: 10.1016/j.ejor.2013.10.015. URL http://linkinghub.elsevier.com/retrieve/pii/S0377221713008382.

[26] Nitish Umang, Michel Bierlaire, and Ilaria Vacca. Exact and heuristic methods to solve the berth allocation problem in bulk ports. *Transportation Research Part E: Logistics and Transportation Review*, 54:14–31, August 2013. ISSN 13665545. doi: 10.1016/j.tre.2013.03.003. URL http://linkinghub.elsevier.com/retrieve/pii/S1366554513000537.

[27] UNCTAD. *Review of Maritime Transport: 2014*. 2015. ISBN 9789211128109.

[28] Ilaria Vacca, Matteo Salani, and Michel Bierlaire. An Exact Algorithm for the Integrated Planning of Berth Allocation and Quay Crane Assignment. *Transportation Science*, 47(2):148–161, May 2013. ISSN 0041-1655. doi: 10.1287/trsc.1120.0428. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0428.

[29] Yannis Vergados, Rowan Van Schaeren, Wout Dullaert, and Birger Raa. The Berth Allocation and Quay Crane Assignment Problem Using a CP Approach. In *Principles and Practice of Constraint Programming*, pages 880–896. Lecture Notes in Computer Science, 2013. URL http://link.springer.com/chapter/10.1007%2F978-3-642-40627-0_64.

[30] Chunxia Yang, Xiaojun Wang, and Zhenfeng Li. An optimization approach for coupling problem of berth allocation and quay crane assignment in container terminal. *Computers & Industrial Engineering*, 63(1):243–253, August 2012. ISSN 03608352. doi: 10.1016/j.cie.2012.03.004. URL http://linkinghub.elsevier.com/retrieve/pii/S0360835212000733.